Centre de recherche sur les transports (C.R.T.)
Centre for Research on Transportation

CRT-2003-14

VEHICLE ROUTING PROBLEM WITH TIME WINDOWS, PART II: METAHEURISTICS

by

Olli Bräysy
Michel Gendreau

June 2003

Université de Montréal
C.P. 6128, succursale Centre-ville, Montréal QC H3C 3J7 Canada
Téléphone: (514) 343-7575 / Télécopieur: (514) 343-7121

crt@crt.umontreal.ca
www.crt.umontreal.ca

HEC ÉCOLE POLYTECHNIQUE MONTRÉAL Université de Montréal

103-07105/

PUBLICATION
CRT-2003-14

VEHICLE ROUTING PROBLEM WITH TIME WINDOWS, PART II: METAHEURISTICS

by

Olli Bräysy[1]
Michel Gendreau[2]

* This research was supported in part by the Emil Aaltonen Foundation, the Natural Sciences and Engineering Research Council of Canada (NSERC), Liikesivistysrahasto, Volvo Foundations and by the TOP project funded by the Research Council of Norway.

[1] SINTEF Applied Mathematics, Department of Optimisation, P.O. Box 124 Blindern, N-0314 Oslo, Norway (Olli.Braysy@math.sintef.no)

[2] Département d'informatique et de recherche opérationnelle and Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7 (michelg@crt.umontreal.ca)

Département d'informatique et de recherche opérationnelle - Publication #1213

Dépôt légal – Bibliothèque nationale du Québec
Bibliothèque nationale du Canada, 2003

Abstract

This paper surveys the research on the Tabu Search heuristics for the Vehicle Routing Problem with Time Windows (VRPTW). The VRPTW can be described as the problem of designing least cost routes for a fleet of vehicles from one depot to a set of geographically scattered points. The routes must be designed in such a way that each point is visited only once by exactly one vehicle within a given time interval; all routes start and end at the depot, and the total demands of all points on one particular route must not exceed the capacity of the vehicle. In addition to describing basic features of each method, experimental results for Solomon's benchmark test problems are presented and analyzed.

Keywords: Metaheuristics, tabu search, vehicle routing, time windows.

Résumé

Cet article présente un survol des algorithmes de recherche avec tabous pour le Problème de Tournées de Véhicules avec Fenêtres de Temps (PTVFT). Le PTVFT consiste à déterminer un ensemble de tournées à coût minimal à partir d'un dépôt vers un ensemble de points dispersés géographiquement. Les tournées doivent être telles que chaque point soit visité par un seul véhicule à l'intérieur d'un intervalle de temps donné; toutes les tournées débutent et se terminent au dépôt et la somme des demandes des points visités par chaque tournée ne doit pas excéder la capacité du véhicule. Les principaux types d'algorithmes évolutionnistes pour le PTVFT sont les algorithmes génétiques et les stratégies d'évolution. En plus de décrire les principaux éléments de chaque méthode, nous présentons et analysons les résultats expérimentaux obtenus pour les problèmes tests de Solomon.

Mots-clés : métaheuristiques, recherche avec tabous, tournées de véhicules, fenêtres de temps.

The motivation and background for researching the Vehicle Routing Problem with Time Windows (VRPTW) as well as the problem description are discussed in the first part of this article (Bräysy and Gendreau, 2002a). However, for the sake of completeness, we recall the basic elements of the VRPTW here. The VRPTW can be described as the problem of designing least cost routes from one depot to a set of geographically scattered points. The routes must be designed in such a way that each point is visited only once by exactly one vehicle within a given time interval; all routes start and end at the depot, and the total demands of all points on one particular route must not exceed the capacity of the vehicle.

The VRPTW has multiple objectives in that the goal is to minimize not only the number of vehicles required, but also the total travel time or total travel distance incurred by the fleet of vehicles. A hierarchical objective function is typically associated with all procedures studied. That is, the number of routes is first minimized and then, for the same number of routes, the total traveled distance or time is minimized. The VRPTW is a basic distribution management problem that can be used to model many real-world problems and it has been the subject of intensive research efforts focused mainly on heuristic and metaheuristic approaches. The heuristic solution methods and previous survey papers are discussed in the first part of this article. Here, we focus on metaheuristic approaches. Metaheuristics are general solution procedures that explore the solution space to identify good solutions and often embed some of the standard route construction and improvement heuristics. In a major departure from classical approaches, metaheuristics allow deteriorating and even infeasible intermediate solutions in the course of the search process. For the most well-known metaheuristic approaches, a description of the basic principles is given first followed by a description of applications to the VRPTW. Most of the methods are compared with other similar approaches based on the experimental results obtained for the Solomon's (1987) test problems.

The remainder of this paper is organized as follows. The Tabu Search algorithms for the VRPTW are reviewed in section 1. Section 2 focuses on genetic algorithms and evolution strategies, as well as hybrids based on them. Other metaheuristic approaches are discussed in section 3, including methods such as Simulated Annealing, Ant Algorithms, Guided Local Search, Variable Neighborhood Search, etc. In section 4, we summarize the findings and analyze the efficiency of the described metaheuristics. Finally, section 5 concludes the paper.

1. TABU SEARCH ALGORITHMS

TABU SEARCH (TS) is a local search metaheuristic introduced by Glover (1986). Details about tabu search can also be found in Glover (1989), Glover (1990), Hertz et al. (1997), Glover and Laguna (1997) and Gendreau (2003). TS explores the solution space by moving at each iteration from a solution s to the best solution in a subset of its neighborhood $N(s)$. Contrary to classical descent methods, the current solution may deteriorate from one iteration to the next. New poorer solutions are accepted only to avoid paths already investigated. This insures new regions of a problem's solution space will be investigated with the goal of avoiding local minima and ultimately finding the desired solution. To avoid cycling, solutions possessing some attributes of recently explored solutions are temporarily declared tabu or forbidden. The duration that an attribute remains tabu is called its tabu-tenure and it can vary over different intervals of time. The tabu status can be overridden if certain conditions are met; this is called the aspiration criterion and it happens, for example, when a tabu solution is better than any previously seen solution. Finally, various techniques are often employed to diversify or to intensify the search process. For theoretical aspects of tabu search, see Faigle and Kern (1992) and Fox (1993).

Garcia et al. (1994) were the first to apply tabu search for VRPTW. The authors presented a parallel implementation on a network of 16 Meiko T-800 transputers. The tabu search they developed is a fairly simple one, involving Solomon's I1 insertion heuristic to create an initial solution and 2-opt* and Or-opt exchanges for improvement (for details, see part I of this survey). Many authors since that time have presented numerous tabu search implementations involving sophisticated diversification and intensification techniques, explicit strategies for minimizing the number of routes, complex post-optimization techniques, hybridizations with other search techniques such as simulated annealing and genetic algorithms, parallel implementations, and allowance of infeasible solutions during the search.

The initial solution is typically created with some cheapest insertion heuristic, described in the first part of this survey article. The most common is Solomon's (1987) I1 insertion heuristic. An exception can be found in Chiang and Russell (1997), where a parallel version of the insertion heuristic of Russell (1995) is used. De Backer and Furnon (1997) and Schulze and Fahle (1999) use the savings heuristic of Clarke and Wright (1964); Tan et al. (2000) use a modified version of Solomon's insertion heuristic, proposed by Thangiah (1994), and Cordeau et al. (2001) use a modified version of the sweep heuristic developed by Gillett and Miller (1974). Lau et al. (2002) introduce a concept of holding list, a data structure containing the unserviced customers. In the beginning all customers are in the holding list, and simple relocate and exchange operators are then used to transfer customers back and forth the holding list.

After creating an initial solution, an attempt is made to improve it using local search with one or more neighborhood structures and the best-accept strategy. Most of the neighborhoods used are well known and were previously introduced in the context of various construction and improvement heuristics. Examples of such neighborhoods are 2-opt, Or-opt, 2-opt*, relocate, exchange and CROSS-GENI-, and λ-exchanges, discussed in detail in the first part of this survey article.

To reduce the complexity of the search, some authors propose special strategies for limiting the neighborhood. For instance, Garcia et al. (1994) only allow moves involving arcs that are close in distance. Taillard et al. (1997) decompose solutions into a collection of disjoint subsets of routes by using the polar angle associated with the center of gravity of each route. Tabu search is then applied to each subset separately. A complete solution is reconstructed by merging the new routes found by tabu search. Another frequently used strategy to speed up the search is to implement the proposed algorithm in parallel on several processors. For instance, Badeau et al. (1997) apply the solution approach of Taillard et al. (1997) using a 2-level parallel implementation. Results on benchmark problems show that this parallelization of the original sequential approach does not degrade solution quality, for the same amount of computation, while providing substantial speed-ups. Other studies describing parallel implementations can be found in Garcia et al. (1994) and Schulze and Fahle (1999). On the other hand, to cross the barriers of the search space, created by time window constraints, some authors allow infeasibilities during the search. For instance, Brandão (1999), Cordeau et al. (2001) and Lau et al. (2002) allow violation of each constraint type (load, duration and time windows constraints). The violations of constraints are penalized in the cost function and the parameter values regarding each type of violation are adjusted dynamically.

Since the number of routes is often considered as the primary objective, some authors use different explicit strategies for reducing the number of routes. For example, the algorithms of Garcia et al. (1994) and Potvin et al. (1996) try to move customers from routes with a few customers into other routes using Or-opt exchanges. Similarly, the method of Schulze and Fahle (1999) tries to eliminate routes having at most three customers by trying to move these customers into other routes. In Lau et al. (2002) a limit is set for the number of routes that cannot be exceeded during the search.

Most of the proposed tabu searches use specialized diversification and intensification strategies to guide the search. For example, Rochat and Taillard (1995) propose using a so-called "adaptive memory". The adaptive memory is a pool of routes taken from the best solutions visited during the search. Its purpose is to provide new starting solutions for the tabu search through selection and combination of routes extracted from the memory. The selection of routes from the memory is done probabilistically and the probability of selecting a particular route depends on the value of the solution

the route belongs to. The selected tours are improved using tabu search and inserted subsequently back into adaptive memory. Later, Taillard et al. (1997) used the same strategy to tackle the VRP with soft time windows. In this problem, lateness at customer locations is allowed although a penalty is incurred and added to the objective value. Taillard et al. (1997) also diversify the search by penalizing frequently performed exchanges and intensify the search by reordering the customers within the best routes using Solomon's I1 insertion heuristic. Chiang and Russell (1997), Schulze and Fahle (1999) and Cordeau et al. (2001) use a similar strategy for diversification, but in Chiang and Russell (1997) the intensification is used to reduce waiting time by forbidding certain customers from moving into another route. Schulze and Fahle (1999) also propose a strategy similar to adaptive memory, wherein all routes generated by the tabu search heuristic are collected in a pool. At the termination of the local optimization steps, the worst solution is replaced by a new one created by solving the set-covering problem on the routes in the pool using the Lagrangian relaxation-based heuristic of Beasley (1990).

Carlton (1995) and Chiang and Russell (1997) test a reactive tabu search that dynamically adjusts its parameter values based on the current search status to avoid both cycles as well as an overly constrained search path. More precisely, the size of the tabu list is managed by increasing the tabu list size if identical solutions occur too often, and reducing it if no feasible solution can be found. Tan et al. (2000) diversify the search each time a local minimum is found by performing a series of random λ-interchange hops combined with the 2-opt* operator. A candidate list is maintained to record elite solutions discovered during the search process. These elite solutions are then used as a starting point for intensification. Lau et al. (2000) present a generic constraint-based diversification technique, where VRPTW is modeled as a linear constraint satisfaction problem that is solved by a simple local search algorithm.

Finally, several authors report using various post-optimization techniques. For instance, Rochat and Taillard (1995) solve exactly a set-partitioning problem at the end, using the routes in the adaptive memory to return the best possible solution. Taillard et al. (1997) apply an adaptation of the GENIUS heuristic (Gendreau et al., 1992) for time windows to each individual route of the final solution. Similarly, in Cordeau et al. (2001) the best solution identified after n iterations is post-optimized by applying to each individual route a specialized heuristic for the Traveling Salesman Problem with Time Windows (Gendreau et al., 1998). The main features of the tabu search heuristics just described are summarized in Table 1, where we present the initial solution heuristics, neighborhood operators used, as well as mention whether the proposed approach uses explicit strategies for reducing the number of routes. In the last column, some notes are given. Further details about tabu search heuristics for VRPTW can be found in Bräysy and Gendreau (2002b).

TABLE 1

The main features of tabu search heuristics for VRPTW.

Authors	Year	Initial solution	Neighborhood Operators	Route min.	Notes
Garcia et al.	1994	Solomon's I1 heuristic	2-opt*, Or-opt	Yes	Neighborhood restricted to arcs close in distance
Rochat et al.	1995	Modification of Solomon's I1, 2-opt	2-opt, relocate	No	Adaptive memory
Carlton	1995	Insertion heuristic	relocate	No	Reactive tabu search
Potvin et al.	1996	Solomon's I1 heuristic	2-opt*, Or-opt	Yes	Neighborhood restricted to arcs close in distance
Taillard et al.	1997	Solomon's I1 heuristic	CROSS	No	Soft time windows, adaptive memory
Badeau et al.	1997	Solomon's I1 heuristic	CROSS	No	Soft time windows, adaptive memory
Chiang et al	1997	Modification of Russell (1995)	λ-interchange	No	Reactive tabu search
De Backer et al.	1997	Savings heuristic	exchange, relocate, 2-opt*, 2-opt, Or-opt	No	Constraint programming used to check feasibility of moves
Brandão	1999	Insertion heuristic	relocate, exchange, GENI	No	Neighborhoods restricted to arcs close in distance
Schulze et al.	1999	Solomon's I1, parallel I1 and savings heuristic	Ejection chains, Or-opt	Yes	Generated routes stored in a pool
Tan et al.	2000	Insertion heuristic of Thangiah (1994)	λ-interchange, 2-opt*	No	———
Lau et al.	2000	Insertion heuristic	exchange, relocate	No	Constraint based diversification
Cordeau et al.	2001	Modification of Sweep heuristic	relocate, GENI	No	———
Lau et al.	2002	Relocation from a holding list	exchange, relocate	Yes	Holding list for unrouted nodes, limit for number of routes

The tabu search algorithms described above are compared in Table 2, where the first column to the left gives the authors. Columns R1, R2, C1, C2, RC1 and RC2 present the average number of vehicles and average total distance with respect to the six problem groups of Solomon (1987). Finally, the rightmost column indicates the cumulative number of vehicles (CNV) and cumulative total distance (CTD) over all 56 test problems. For more information about Solomon's benchmark problems, we refer to the first part of this article and the original paper by Solomon (1987). Due to the lack of exact information, we cannot consider all algorithms here. Table 2 shows the best solutions attained with each method without paying attention to the computational effort. Even though Brandão (1999) uses rounded distances during the execution of the algorithm, we believe that the differences in final solutions remain small and the results are therefore comparable. To our knowledge, only De Backer and Furnon (1997) proposed a deterministic method. All other procedures in Table 2 are stochastic, i.e., in practice one gets different results with each run. All methods consider the number of vehicles as the primary optimization criterion. The only exceptions are the approaches of De Backer and Furnon (1997) and Tan et al. (2000) that concentrate solely on minimization of distance. The second objective is total traveled distance in Rochat and Taillard (1995), Taillard et al. (1997), Chiang and Russell

(1997), Brandão (1999), Cordeau et al. (2001) and Lau et al. (2000 and 2002). The other procedures use total duration of routes as the second objective, causing a slight over-estimation of the reported total distance values. The CTD values in Tables 1, 2 and 3 are rounded to integers due to the usage of rounded distance measures reported by other authors for calculation.

TABLE 2

Comparison of tabu search algorithms. For each algorithm the average results with respect to Solomon's benchmarks are depicted. The notations CNV and CTD in the rightmost column indicate the cumulative number of vehicles and cumulative total distance over all 56 test problems.

Author	R1	R2	C1	C2	RC1	RC2	CNV/CTD
Garcia et al. (1994)	**12.92**	**3.09**	**10.00**	**3.00**	**12.88**	**3.75**	**436**
	1317.7	1222.6	877.1	602.3	1473.5	1527.0	65977
Rochat et al. (1995)	**12.25**	**2.91**	**10.00**	**3.00**	**11.88**	**3.38**	**415**
	1208.50	961.72	828.38	589.86	1377.39	1119.59	57231
Potvin et al. (1996)	**12.50**	**3.09**	**10.00**	**3.00**	**12.63**	**3.38**	**426**
	1294.5	1154.4	850.2	594.6	1456.3	1404.8	63530
Taillard et al. (1997)	**12.17**	**2.82**	**10.00**	**3.00**	**11.50**	**3.38**	**410**
	1209.35	980.27	828.38	589.86	1389.22	1117.44	57523
Chiang et al. (1997)	**12.17**	**2.73**	**10.00**	**3.00**	**11.88**	**3.25**	**411**
	1204.19	986.32	828.38	591.42	1397.44	1229.54	58502
De Backer et al. (1997)	**14.17**	**5.27**	**10.00**	**3.25**	**14.25**	**6.25**	**508**
	1214.86	930.18	829.77	604.84	1385.12	1099.96	56998
Brandão (1999)	**12.58**	**3.18**	**10.00**	**3.00**	**12.13**	**3.50**	**425**
	1205	995	829	591	1371	1250	58562
Schulze et al. (1999)	**12.25**	**2.82**	**10.00**	**3.00**	**11.75**	**3.38**	**414**
	1239.15	1066.68	828.94	589.93	1409.26	1286.05	60346
Tan et al. (2000)	**13.83**	**3.82**	**10.00**	**3.25**	**13.63**	**4.25**	**467**
	1266.37	1080.24	870.87	634.85	1458.16	1293.38	62008
Lau et al. (2000)	**14.00**	**3.55**	**10.00**	**3.00**	**13.63**	**4.25**	**464**
	1211.54	960.43	832.13	612.25	1385.05	1232.65	58432
Cordeau et al. (2001)	**12.08**	**2.73**	**10.00**	**3.00**	**11.50**	**3.25**	**407**
	1210.14	969.57	828.38	589.86	1389.78	1134.52	57556
Lau et al. (2002)	**12.17**	**3.00**	**10.00**	**3.00**	**12.25**	**3.38**	**418**
	1211.55	1001.12	832.13	589.86	1418.77	1170.93	58477

According to Table 2, the tabu search by Cordeau et al. (2001) seems to produce the best results in terms of solution quality. However, the difference with regard to other well-performing approaches by Taillard et al. (1997) and Chiang and Russell (1997) is less than 1% in the number of vehicles. Regarding the total traveled distance, the differences between the three methods are also small. The differences in CTD remain within 2%. The algorithm by De Backer and Furnon (1997) seems to give the worst results with respect to the number of vehicles. The reason for this can be found in the optimization criterion used. De Backer and Furnon (1997) and Tan et al. (2000) consider only the total traveled distance, while the other procedures minimize the number of vehicles first. However, in spite of this difference in objective function, the method by De Backer and Furnon (1997) produces better

outcomes than the other approaches in terms of total distance only for problem groups R2 and RC2. Overall, the difference in cumulative number of vehicles is about 14% if De Backer and Furnon (1997) and Tan et al. (2000) are not considered. In our opinion, this difference is quite significant, and in terms of total distance, the differences are even greater. For example, the difference between the approaches of Garcia et al. (1994) and Rochat and Taillard (1995) in CTD is about 15% and the difference between Garcia et al. (1994) and Taillard et al. (1997) in problem group RC2 is about 37%.

As far as the computational effort is concerned, conclusions are very difficult to draw, since most of the authors do not report the CPU time consumption or the number of runs used to obtain the results in Table 2. For example, it is impossible to compare the best approaches by Cordeau et al. (2001), Taillard et al. (1997) and Chiang and Russell (1997) in terms of CPU time consumption. Another comparison with other metaheuristics is presented in Table 6 and Figure 1, where only results for which computational effort is reported are considered.

2. GENETIC ALGORITHMS

THE GENETIC Algorithm (GA) is an adaptive heuristic search method based on population genetics. The basic concepts were developed by Holland (1975), while the practicality of using the GA to solve complex problems was demonstrated in De Jong (1975) and Goldberg (1989). Details and references about genetic algorithms can also be found, for example, in Mühlenbein (1997) and Alander (2000) respectively. GA evolves a population of individuals encoded as chromosomes by creating new generations of offspring through an iterative process until some convergence criteria are met. Such criteria might, for instance, refer to a maximum number of generations, or the convergence to a homogeneous population composed of similar individuals. The best chromosome generated is then decoded, providing the corresponding solution.

The creation of a new generation of individuals involves four major steps or phases: representation, selection, recombination and mutation. The representation of the solution space consists of encoding significant features of a solution as a chromosome, defining an individual member of a population. The selection phase consist of randomly choosing two parent individuals from the population for mating purposes. The probability of selecting a population member is generally proportional to its fitness in order to emphasize genetic quality while maintaining genetic diversity. Here, fitness refers to a measure of profit, utility or goodness to be maximized while exploring the solution space. The recombination or reproduction process makes use of genes of selected parents to produce offspring that will form the next generation. As for mutation, it consists of randomly modifying some gene(s) of a single individual at a time to further explore the solution space and ensure, or preserve, genetic diversity. The occurrence of mutation is generally associated with a low probability. A new generation

is created by repeating the selection, reproduction and mutation processes until a speficied set of new chromosomes have been created and placed in the new population. The set of chromosomes to be created and replaced depends on the selection strategy and type of GA applied. In some cases, all chromosomes in the old population are replaced by new ones, and in some cases a set of old chromosomes are preserved. A proper balance between genetic quality and diversity is therefore required within the population in order to support efficient search.

Thangiah et al. (1991) were the first to apply a genetic algorithm to VRPTW (the same method is described in more detail in Thangiah, 1995a, and in the following we refer only to the latter paper). This first paper describes an approach that uses a genetic algorithm to find good clusters of customers, within a "cluster first, route second" problem-solving strategy. The routes within each cluster are then constructed with a cheapest insertion heuristics, and also λ–exchanges are applied to improve solution quality. During the past few years, numerous papers have been written on generating good solutions for VRPTW with GAs. Almost all these papers present hybridizations of a GA with different construction heuristics (Blanton and Wainwright, 1993 and Berger et al., 1998), local searches (Thangiah 1995a and 1995b; Thangiah et al., 1995; Potvin and Bengio, 1996 and Jung and Moon, 2002) and other metaheuristics such as tabu search (Wee Kit 2001) and ant colony system (Berger et al., 2001).

Homberger and Gehring (1999) present two evolution strategies (Rechenberg, 1973 and Schwefel, 1977) for the VRPTW. Together with GAs and evolutionary programming, the evolution strategies form the class of evolutionary algorithms (Fogel, 1995). By definition, the main differences between these three types of algorithms lie in the representation and in the role of mutation. For more details, we refer the reader to Bräysy and Gendreau (2002c). In evolution strategies of Homberger and Gehring (1999) the individual representation includes a vector of so-called "strategy parameters" in addition to the solution vector and both components are evolved by means of recombination and mutation operators. In the proposed application to the VRPTW, these strategy parameters refer to how often a randomly selected local search operator is applied, and to a binary parameter used to alternate the search between minimizing the number of vehicles and total distance. Only one offspring is created through the recombination of parents. In this way, a number $\lambda > \mu$ of offspring is created, where μ is the population size. At the end, fitness values are used to select μ offspring for the next population.

In Gehring and Homberger (1999) the evolution strategies of Homberger and Gehring (1999) is hybridized with tabu search to minimize the total distance, and the approach is parallelized using the concept of cooperative autonomy, i.e., several autonomous sequential solution procedures cooperate through the exchange of solutions. The authors also develop a new set of larger benchmark problems that are based on the benchmark problems of Solomon (1987). Gehring and Homberger (2001)

introduce three different improvements to the parallel method of Gehring and Homberger (1999). In the evaluation of individuals, capacity related information is also used to determine the routes for elimination. Additional improvements include greater population size and new termination criteria. In Homberger and Gehring (2002), a single processor implementation of Gehring and Homberger (2001) is presented. Another difference is that in Homberger and Gehring (2002) capacity information is not used in the evaluation criterion. Mester (2002) has also experimented with evolution strategies similar to Homberger and Gehring (1999). Le Bouthillier and Crainic (2002) present a parallel co-operative methodology in which several agents communicate through a pool of feasible solutions. The agents consist of simple construction and local search algorithms, GAs and adaptations of TABUROUTE method of Gendreau et al. (1994).

Although theoretical results that characterize the behavior of the GA have been obtained for bit-string chromosomes, not all problems lend themselves easily to this representation. This is the case, in particular, for sequencing problems, such as the vehicle routing problem, where an integer representation is more often appropriate. Therefore, in most applications to VRPTW, the genetic operators are applied directly to solutions, represented as integer strings, thus avoiding coding issues. In most cases the authors use delimiters to separate customers served by different routes. An exception is found in Tan et al. (2001a), where the basic grouping is determined by the insertion heuristic of Solomon (1987), and λ-interchanges are used to create alternative groupings. A similar study is reported also in Tan et al. (2001c). Jung and Moon (2002) suggest using the 2D image of a solution for chromosomal cutting. In Thangiah (1995a and 1995b) and Thangiah et al. (1995), traditional bit-string encoding is used, and each chromosome represents a set of possible clustering schemes within a cluster-first, route-second search strategy. Blanton and Wainwright (1993) used the so-called Davis encoding method, where a chromosome represents a permutation of n customers to be partitioned into m vehicles. The first m customers of a chromosome are placed into the m different vehicles. The remaining $n - m$ customers are examined individually using a greedy insertion heuristic. In the messy genetic algorithm (Goldberg et al., 1989) of Tan et al. (2001b) the solution is encoded using ordered pairs, consisting of customer and vehicle identification indexes.

The initial population is typically created either randomly or using modifications of well-known construction heuristics. A random strategy can be found in Blanton et al. (1993), Tan et al. (2001b) and Le Bouthillier and Crainic (2002), though the last one applies also a set of construction heuristics combined with 2-opt, 3-opt and Or-opt improvement heuristics. Thangiah (1995a and 1995b) and Thangiah et al. (1995) cluster the customers randomly into separate groups, and then use the cheapest insertion heuristic of Golden and Stewart (1985) to route customers within each group. Homberger and

Gehring (1999 and 2002) and Gehring and Homberger (1999 and 2001) use a modification of the savings heuristic of Clarke and Wright (1964), where the savings element is selected randomly from the savings list. Berger et al. (2001) modify a randomly generated initial population with λ-exchanges and a reinitialization procedure based on the insertion procedure of Liu and Shen (1999), in order to create a population of solutions with the number of vehicles equal to the lowest found. In Mester (2002) all customers are first served by separate routes. Then, a set of six initial solutions is created using cheapest reinsertions of single customers with varying insertion criteria, and the best solution obtained is selected as starting point. Solomon's insertion heuristic is used in Potvin and Bengio (1996), Tan et al. (2001a) and Jung and Moon (2002). The last two authors also create a set of solutions randomly and by modifying the heuristic solution with λ-interchanges. Berger et al. (1998) use Solomon's (1987) nearest neighbor heuristic. To the best of our knowledge, only Berger et al. (1998) and Berger et al. (2001) use more than one population. For instance, the algorithm proposed in Berger et al. (2001) evolves two populations in parallel. The first population is used to minimize the total distance and the second population tries to minimize violations of the time window constraints.

Fitness values are usually based on routing costs, i.e., number of routes, total distance and duration. Le Bouthillier and Crainic (2002) consider in addition waiting and residual time at each customer. In Blanton and Wainwright (1993) the fitness value is the number of unserviced customers in case of infeasible solutions. Homberger and Gehring (1999 and 2002) and Gehring and Homberger (1999 and 2001) also consider how easily the shortest route of the solution (in terms of the number of customers on the route) can be eliminated, in addition to the number of routes and the total distance. In Berger et al. (2001), the evaluation of the individuals is based on a weighted sum of objectives related to violated constraints, number of vehicles and total distance.

The most typical selection scheme for selecting a pair of individuals (parents) for recombination is the well-known roulette-wheel scheme. In this stochastic scheme, the probability of selecting an individual is proportional to its fitness value. For details, see Goldberg (1989). Tan et al. (2001a) and Jung and Moon (2002) use so-called tournament selection. The basic idea is to perform the roulette-wheel scheme twice and to select the better out of the two individuals identified by the roulette-wheel scheme. In Wee Kit et al. (2001), Homberger and Gehring (1999 and 2002), Gehring and Homberger (1999 and 2001) and Mester (2002) the parents are selected randomly. Finally, Potvin and Bengio (1996) and Le Bouthillier and Crainic (2002) use a ranking scheme, where the probability of selecting an individual is based on its rank.

The recombination is the most crucial part of a genetic algorithm. The traditional 2-point crossover, which exchanges a randomly selected portion of the bit string between the chromosomes, is used in

Thangiah (1995a and 1995b) and Thangiah et al. (1995), while Tan et al. (2001a) and Tan et al. (2001b) use the well-known PMX and 1-point crossovers, respectively. The basic idea in PMX crossover is to choose two cut points at random, and based on these cut points, to perform a series of swapping operations in the second parent. The 1-point crossover switches two sets of customers to be serviced by two different routes. Traditional order-based operators, based on a precedence relationship among the genes in a chromosome, are used in Blanton and Wainwright (1993), Homberger and Gehring (1999) and Le Bouthillier and Crainic (2002). The last authors use also well-known edge recombination crossover, and in Homberger and Gehring (1999) crossover is used to modify the initially randomly created mutation codes. The mutation code is used to control a set of removal and insertion operators performed by the Or-opt operator.

In the context of VRPTW, many authors have proposed specialized heuristic crossover procedures, instead of traditional operators. Potvin and Bengio (1996) propose a sequence-based and a route-based crossover. The sequence-based crossover first selects a link randomly from each parent solution. Then, the customers that are serviced before the breakpoint on the route of parent-solution P_1 are linked to the customers that are serviced after the breakpoint on the route of parent solution P_2. Finally, the new route replaces the old one in parent solution P_1. The route-based crossover replaces one route of parent solution P_2 by a route of parent solution P_1. In Berger et al. (1998) and Berger et al. (2001), a removal procedure is first carried out to remove some key customer nodes in a similar fashion to Large Neighborhood Search (LNS) of Shaw (1998). Then, an insertion procedure (inspired from Solomon (1987) and Liu and Shen (1999), respectively) is locally applied to reconstruct the partial solution. The first operator of Wee Kit et al. (2001) tries to modify the order of the customers in the first parent by trying to create consecutive pairs of customers according to the second parent. The second crossover operator tries to copy common characteristics of parent solutions to offspring by modifying the seed selection procedure and cost function of an insertion heuristic similar to Solomon's (1987). In Jung and Moon (2002) the recombination is based on selecting a set of inherited arcs using different types of curves drawn on the 2D space where customers are located, and including the missing arcs in nearest neighbor manner.

The mutation is often considered as secondary strategy, and its purpose in traditional genetic algorithms is mainly to help escape from local minima. However, in the evolution strategies of Homberger and Gehring (1999 and 2002), Gehring and Homberger (1999 and 2001) and Mester (2002), the search is mainly driven by mutation, based on traditional local search operators (2-opt*, Or-opt and λ-interchanges). Mester (2002) uses also GENIUS heuristic of Gendreau et al. (1992) and so called multi-parametric mutation that consists of removing a set of customers from a solution

randomly, based on the distance to the depot or by selecting one customer from each route. Then, a cheapest insertion heuristic is used to reschedule the removed customers. In Potvin and Bengio (1996), Wee Kit et al. (2001), Le Bouthillier and Crainic (2002) and Jung and Moon (2002) the mutation is entirely or partially based on well-known local search operators (Or-opt, crossover and relocation). In Potvin and Bengio (1996), Berger et al. (1998), Homberger and Gehring (1999 and 2002) and Gehring and Homberger (1999 and 2001) and Berger et al. (2001), mutation is also used to reduce the number of routes, by using Or-opt, Solomon's insertion heuristic or by performing one or several subsequent relocate moves. Berger et al. (1998) use mutation to locally reorder routes with the Nearest Neighbor heuristic of Solomon (1987).

Berger et al. (2001) present five mutation operators including the LNS of Shaw (1998), λ-exchanges, exchange of customers served too late in the current solution, elimination of the shortest route using the procedure by Liu and Shen (1999) and within-route reordering using Solomon's (1987) heuristic.

TABLE 3

The main features of genetic algorithms and evolution strategies for the VRPTW.

Authors	Year	Initial population	Crossover	Mutation
Blanton et al.	1993	Random ordering	Special order-based operators	Random exchange of two genes
Thangiah	1995a	Random clustering + insertion heuristic	2-point crossover	Random change of bit values
Thangiah	1995b	Random clustering + insertion heuristic	2-point crossover	Random change of bit values
Thangiah et al.	1995	Random clustering + insertion heuristic	2-point crossover	Random change of bit values
Potvin et al.	1996	Solomon's insertion	Reinsertion of a route or route segment into another parent	Combinations of relocate operators to eliminate routes, and Or-opt
Berger et al.	1998	Nearest neighbor	Modification of LNS of Shaw (1998), reinsertion with modified Solomon's heuristic	Relocate to reduce the number of routes and nearest neighbor for within-route reordering
Homberger et al.	1999	Stochastic savings heuristic	Uniform order-based to create sequence for controlling Or-opt	Or-opt, 2-opt*, λ-interchanges, special Or-opt for route elimination
Gehring et al.	1999	Stochastic savings heuristic	———	Or-opt, 2-opt*, λ-interchanges, special Or-opt for route elimination
Gehring et al.	2001	Stochastic savings heuristic	———	Or-opt, 2-opt*, λ-interchanges, special Or-opt for route elimination
Tan et al.	2001a	Solomon's insertion, λ-interchange, random	PMX	Random swap of nodes
Tan et al.	2001b	Random ordering	1-point	———
Wee Kit et al.	2001	Not defined	Relocations based on second parent, modifies seed selection and cost function of Solomon's I1	Tabu search using 2-opt*, exchange, relocate and 2-opt, applied only later generations

Berger et al.	2001	Random insertion heuristic	Modification of LNS of Shaw (1998), reinsertion with modified Solomon's heuristic and procedure of Liu et al. (1999)	Modified LNS, λ-interchanges, relocate, insertion heuristics of Liu et al. and Solomon
Le Bouthillier et al.	2002	Construction heuristics combined with 2-opt, 3-opt and Or-opt	Order (OX) and edge recombination (ER)	2-opt, Or-opt, 3-opt, TABUROUTE
Mester	2002	Cheapest insertion with varying criteria	———	Or-opt, 2-opt*, λ-interchanges, GENIUS, modified LNS
Jung et al.	2002	Solomon's insertion	Selecting arcs based on 2D image of a solution and nearest neighbor rule	Or-opt, 2-opt*, relocation, splitting of routes
Homberger et al.	2002	Stochastic savings heuristic	———	Or-opt, 2-opt*, λ-interchanges, special Or-opt for route elimination

In some recent papers, different intensification techniques are coupled to a GA. For instance, Tan et al. (2001a) introduce a special hill-climbing technique, wherein a randomly selected part of the population is improved by partial λ-exchanges. In Wee Kit et al. (2001), a simple tabu search based on 2-opt*, exchange, relocate and 2-opt neighborhoods is applied to individual solutions in the later generations in order to intensify the search. Like the tabu searches discussed in section 1, many genetic algorithms allow infeasibilities during the search in order to escape from local minima. Examples of such strategies can be found in Blanton and Wainwright (1993), Thangiah (1995a and 1995b), Thangiah et al. (1995), Berger et al. (2001) and Le Bouthillier and Crainic (2002). Parallel implementations can be found in Gehring and Homberger (1999 and 2001), Le Bouthillier and Crainic (2002) and Homberger and Gehring (2002). Mester (2002) proposes a set of strategies for dividing a problem into parts to speed up the search. The main features of the various genetic algorithms and evolution strategies described above are summarized in Table 3, where we report the authors, strategies used to create the initial population, as well as crossover and mutation operators used. Further details about genetic algorithms and evolution strategies for the VRPTW can be found in Bräysy and Gendreau (2002c).

The genetic algorithms and evolution strategies described above are compared in Table 4. First, the best results averaged over each problem set of Solomon (1987) are reported. The latter part of the Table describes the computer used, number of independent runs and average time consumption in minutes as reported by the authors. All algorithms, in Table 4 are stochastic and they are implemented in C, except Wee Kit et al. (2001) and Mester (2002) that are coded in Java and Visual Basic, respectively. A hierarchical objective function is used in every case, except in Tan et al. (2001a and 2001b) and Jung and Moon (2002), where the only objective is to minimize total distance. The number of routes is considered as the primary objective and, for the same number of routes, the secondary

objective is to minimize the total traveled distance. An exception is found in Potvin and Bengio (1996), where the second objective is to minimize the total duration of routes. This may cause some over-estimation of traveled distance that should be taken into account when comparing the total distance values.

According to Table 4, the methods by Homberger and Gehring (2002), Berger et al. (2001) and Mester (2002) seem to produce the best results. The differences between these best methods in terms of solution quality are small. When it comes to other approaches, the worst results regarding the CNV are produced by Tan et al. (2000 and 2001a) and Jung and Moon (2002) that focus only on minimizing the total distance. Jung and Moon (2002) seems to be clearly the best of the three methods, producing results that are competitive even with the best known in terms of distance. Problem group RC2 seems to be the most problematic regarding the total traveled distance. The difference between Thangiah (1995a) and Gehring and Homberger (2001) is about 25%, which can hardly be justified in practical settings.

TABLE 4

Comparison of evolutionary and genetic algorithms. For each algorithm, the average results with respect to Solomon's benchmarks are reported. Notations CNV and CTD in the rightmost column indicate the cumulative number of vehicles and cumulative total distance over all 56 test problems.

Author	R1	R2	C1	C2	RC1	RC2	CNV/CTD
(1) Thangiah (1995a)	**12.75**	**3.18**	**10.00**	**3.00**	**12.50**	**3.38**	**429**
	1300.25	1124.28	892.11	749.13	1474.13	1411.13	65074
(2) Potvin et al. (1996)	**12.58**	**3.00**	**10.00**	**3.00**	**12.13**	**3.38**	**422**
	1296.83	1117.64	838.11	590.00	1446.25	1368.13	62634
(3) Berger et al. (1998)	**12.58**	**3.09**	**10.00**	**3.00**	**12.13**	**3.50**	**424**
	1261.58	1030.01	834.61	594.25	1441.35	1284.25	60539
(4) Homberger et al. (1999)	**11.92**	**2.73**	**10.00**	**3.00**	**11.63**	**3.25**	**406**
	1228.06	969.95	828.38	589.86	1392.57	1144.43	57876
(5) Gehring et al. (1999)	**12.42**	**2.82**	**10.00**	**3.00**	**11.88**	**3.25**	**415**
	1198	947	829	590	1356	1140	56942
(6) Gehring et al. (2001)	**12.00**	**2.73**	**10.00**	**3.00**	**11.50**	**3.25**	**406**
	1217.57	961.29	828.63	590.33	1395.13	1139.37	57641
(7) Berger et al. (2001)	**11.92**	**2.73**	**10.00**	**3.00**	**11.50**	**3.25**	**405**
	1221.10	975.43	828.48	589.93	1389.89	1159.37	57952
(8) Tan et al. (2001a)	**13.17**	**5.00**	**10.11**	**3.25**	**13.50**	**5.00**	**478**
	1227	980	861	619	1427	1123	58605
(9) Tan et al. (2001b)	**12.91**	**5.00**	**10.00**	**3.00**	**12.60**	**5.80**	**471**
	1205.0	929.6	841.96	611.2	1392.3	1080.1	56931
(10) Wee Kit et al.(2001)	**12.58**	**3.18**	**10.00**	**3.00**	**12.75**	**3.75**	**432**
	1203.32	951.17	833.32	593.00	1382.06	1132.79	57265
(11) Mester (2002)	**12.00**	**2.73**	**10.00**	**3.00**	**11.50**	**3.25**	**406**
	1208	954	829	590	1387	1119	57219
(12) Jung et al. (2002)	**13.25**	**5.36**	**10.00**	**3.00**	**13.00**	**6.25**	**486**
	1179.95	878.41	828.38	589.86	1343.64	1004.21	54779
(13) Le Bouthillier et al. (2002)	**12.17**	**2.82**	**10.00**	**3.00**	**11.50**	**3.25**	**409**
	1209.27	965.91	828.38	589.86	1389.22	1143.70	57574
(14) Homberger et al. (2002)	**11.92**	**2.73**	**10.00**	**3.00**	**11.50**	**3.25**	**405**

	1212.73	955.03	828.38	589.86	1386.44	1123.17	57309

(1) Solbourne 5/802, –, 2.1 min., (2) Sun Sparc 10, –, 25 min., (3) Sun Sparc 10, –, 1–10 min., (4) Pentium 200 MHz, 10 runs, 13 min., (5) 4×Pentium 200 MHz, 1 run, 10 min., (6) 4×Pentium 400 MHz, 5 runs, 13.5 min., (7) Pentium 400 MHz, –, 30 min., (8) Pentium II 330 MHz, –, 25 min., (9) Pentium II 330 MHz, –, 25 min., (10) Digital Personal Workstation 433a, –, 147.4 min., (11) Pentium III 450 MHz, –, 150.2 min., (12) Pentium III 1 GHz, 100 runs, 0.8 min., (13) 5×Pentium 500 MHz, –, 60 min., (14) Pentium 400 MHz, –,–.

Since only Homberger and Gehring (1999), Gehring and Homberger (1999 and 2001) and Jung and Moon (2002) report the number of runs required to obtain the results in Table 4, it is impossible to draw any final conclusions regarding which method performs best. Considering only these best reported results, methods proposed in Homberger and Gehring (1999 and 2002), and Gehring and Homberger (1999 and 2001) can be considered to be Pareto optimal in terms of solution quality and time consumption. Another comparison with other metaheuristics can be found in Table 6, where only results for which computational effort is reported, are considered.

3. MISCELLANEOUS METAHEURISTICS

IN ADDITION to tabu search and genetic algorithms, a variety of other metaheuristics have been applied to the VRPTW. We now rapidly describe their most important features.

Kontoravdis and Bard (1995) propose a two-phase greedy randomized adaptive search procedure (GRASP) for the VRPTW. The construction procedure first initializes a number of routes by selecting seed customers that are either geographically most dispersed or the most time constrained. After initialization, the algorithm finds the best feasible insertion location in each route for every unrouted customer and calculates a specific penalty value using the Solomon's (1987) cost function. This penalty is the sum of differences between the least insertion cost for each route and the overall best cost. A list L of unassigned customers with the largest penalty value is created and the next customer to be routed is randomly selected from this list. Then local search is applied to the best feasible solution found after every five iterations. In the local search phase, each route is considered for elimination, with routes having fewer customers examined first. If it is not possible to insert a customer into another route, some customer is first removed from the target route r and inserted into another route r', before inserting the current customer into the target route r. Finally, a 2-opt exchange procedure is applied after successful eliminations to improve the solution in terms of distance. To estimate the number of routes, the authors present three lower bounds. The first stems from the underlying Bin Packing structure created by the capacity constraints. The second is derived from the maximum clique of the associated incompatibility graph. An arc in this graph corresponds to a pair of customers who cannot be on the same route due to capacity or time window violations. The third also considers the bin packing problem generated by the time window constraints. Here, bin capacity is the length of the

scheduling horizon, while the items are of a size equal to either the time needed to go from a customer to its closest neighbor or the depot, or the time from the depot to the first customer on each route.

Potvin and Robillard (1995) combine the parallel cheapest insertion heuristic of Potvin and Rousseau (1993) with a competitive neural network. The neural network procedure is used to select the seed customers for the insertion heuristic. The theory of neural networks is beyond the scope of this paper, for details see, for example, Hopfield and Tank (1985) and Kohonen (1988). A weight vector is defined for every vehicle. Initially all weight vectors are placed close to the depot in random fashion. Then one customer at a time is selected and the distance to all weight vectors is calculated. The closest weight vector is updated by moving it closer to the customer. This process is repeated for all customers a number of times; each time the process is restarted, the update of the weight vector becomes less sensitive. At the end of this phase, the seed customers are selected as the customers closest to the weight vectors.

Potvin, Dubé and Robillard (1996) use the same approach as Potvin and Robillard (1995) to select the seed customers for the parallel insertion heuristic of Potvin and Rousseau (1993). The algorithm requires a value for three parameters, α_1, α_2 and μ. The first two constants determine the importance of distance and travel time in the cost function for each unrouted customer. The third factor is used to control the savings in distance. A genetic algorithm is used to find values for these three constants. A stochastic selection procedure is applied to the fitness values based on the number of routes and total route time of the best solution produced by the parallel insertion heuristic. A classical 2-point crossover operator is used for recombination. It swaps a segment of consecutive bits between the parents. The mutation changes with very low probability a bit value from 0 to 1 or from 1 to 0. The results are slightly better compared to using the original insertion heuristic without preprocessing.

Benyahia and Potvin (1995) use a similar GA approach to optimize the parameter values of the sequential and parallel versions of Solomon's (1987) insertion heuristic. However, here seed customers are selected as in original papers instead of neural networks. Moreover, authors introduce additional cost measures, involving slack and waiting times, savings of insertion compared to servicing the customer by individual route, and ratio of additional distance to original distance between the pair of consecutive customers.

Bachem et al. (1996) describe an improvement heuristic based on the mechanisms of trading. The partition of customers into the tours is determined by finding matches in a leveled bipartite graph that the authors call a "trading graph". The nodes correspond to either an insertion (buy) of a customer into a tour or a deletion (sell). The edges represent possible exchanges and the weight of each edge is the gain that is obtained by the corresponding action. Thus, every matching of the trading graph

corresponds to a number of interchanges of customers. In each iteration, tours are shuffled by choosing some permutation at random. Then for each tour either a sell or buy action is selected and finally possible trading matches are evaluated and the best one selected. The approach allows infeasibilities against certain penalty factors, as well as trading matchings with negative weights causing deterioration. Because of this deterioration a tabu list is also added to prevent cycling. The approach was implemented using two different kinds of parallelizations. In the first approach, each tour was mapped into one processor that makes sell or buy decision. To reduce the idle time of the processors, the second approach partitions the current tour plan such that each processor gets about the same number of different tours.

Chiang and Russell (1996) develop a simulated annealing approach for VRPTW. Simulated Annealing (SA) is a stochastic relaxation technique, which has its origin in statistical mechanics. It is based on an analogy from the annealing process of solids, where a solid is heated to a high temperature and gradually cooled in order for it to crystallize in a low energy configuration.

Simulated annealing guides the original local search method in the following way. The solution S' is accepted as the new current solution if $\Delta \leq 0$, where $\Delta = C(S') - C(S)$. To allow the search to escape a local optimum, moves that increase the objective function value are accepted with a probability $e^{-\Delta/T}$ if $\Delta > 0$, where T is a parameter called the "temperature". The value of T varies from a relatively large value to a small value close to zero. These values are controlled by a cooling schedule, which specifies the initial, and temperature values at each stage of the algorithm. For details, see Metropolis et al. (1953), Kirkpatrick et al. (1983) and Aarts et al. (1997).

The authors combine the simulated annealing process with the parallel construction approach of Russell (1995) that incorporates improvement procedures during the construction process. Two different neighborhood structures, the λ-interchange mechanism ($\lambda = 1$) and the k-node interchange process of Christofides and Beasley (1984) are implemented using the first-accept strategy. The enhancement of the annealing process with a short-term memory function via a dynamically varying tabu list is examined as a basis for improving the metaheuristic approach.

Thangiah et al. (1994) develop a number of metaheuristics based on a two-phase approach. In the first phase, an initial solution is created by either the cheapest insertion heuristic or the sectoring based genetic algorithm GIDEON by Thangiah (1995a). The second phase applies one of the following search procedures which use the λ-interchange mechanism: a local search descent procedure using either first best or a global best strategy, a hybrid Simulated Annealing (SA) algorithm with non-monotonic cooling schedule or a hybrid simulated annealing and tabu search. Tabu search algorithm is combined with the SA based acceptance criterion to decide which moves to accept from the candidate

list. The main1p feature of the local search procedures is that infeasible solutions with penalties are allowed if considered attractive.

Tan et al. (2000) develop a fast simulated annealing method based on 2-interchanges with best-accept-strategy and a monotonously decreasing cooling scheme. After the final temperature is reached, special temperature resets based on the initial temperature and the temperature that produced the current best solution are used to restart the procedure. The initial solution is created using a modification of the push-forward insertion heuristic proposed by Thangiah et al. (1994).

Li et al. (2001) propose a tabu-embedded simulated annealing restart metaheuristic. Initial solutions are created by the insertion and extended sweep heuristics of Solomon (1987). Three neighborhood operators based on shifting and exchanging customer segments between and within routes are combined with a simulated annealing procedure that is forced to restart from the current best solution several times. Solomon's insertion procedure is used to reduce the number of routes by trying to insert customers into other routes and to intensify the search by reordering routes. Finally, the search is diversified by performing some random shifts and exchanges of customer segments.

Tan et al. (2001c) hybridize a basic simulated annealing with the tabu search of Tan et al. (2000). The initial solution is created with Solomon's I1 insertion heuristic, and the neighborhood is searched with λ–exchanges using the first accept strategy. A linear cooling schedule is used, and the search is diversified by randomly shifting and interchanging customers between randomly selected routes.

Bent and Van Hentenryck (2001) present a two stage hybrid metaheuristic, where in the first stage a basic simulated annealing is used to minimize the number of routes, and the second stage focuses on distance minimization using the Large Neighborhood Search (Shaw, 1998). The simulated annealing uses randomly the traditional move operators: 2-opt, Or-opt, relocation, exchange and 2-opt*, described in the first part of this survey, and a special evaluation criteria for minimizing the number of routes. In addition to route size and minimal delay, introduced in Homberger and Gehring (1999), the sum of squares of route sizes is used to favor inserting customers from short to larger routes. The Large Neighborhood Search implementation differs from Shaw (1998), described in the first part of this survey, in including a restarting strategy and a more precise lower bound. The initial solution procedure is not described in the paper.

Czech and Czarnas (2002) describe a parallel simulated annealing to find the best possible solutions to a set of Solomon's benchmark instances. The best solutions reported in earlier studies are taken as initial solutions, and the neigborhood search is based on random relocations of single customers with the best-accept strategy. The temperature values are reduced geometrically, and the procedure memorizes the best solutions found during the entire search. Each of the parallel processes carry out

the annealing searches using the same initial solution and cooling schedule, and the processes cooperate at certain intervals by passing their best solutions.

Kilby et al. (1999) introduced Guided Local Search (GLS) for VRPTW. GLS is a memory-based technique developed by Voudouris (1997) and Voudouris and Tsang (1998). It operates by augmenting the cost function with a penalty term based on how close the search moves to previously visited local minima, thus encouraging diversification. GLS moves out of local minima by penalizing particular solution features (usually one) it considers should not occur in a near-optimal solution weighted by the number of times the feature has already been penalized. The more often a feature appears and is penalized, the less likely it is to be penalized further. The authors choose arcs as the feature to penalize. In the initial solution, no visits are allocated to any vehicle. A penalty is associated with not performing a visit, and so the search process constructs a solution in the process of minimizing cost using four different local searches. The local search operators used are 2-opt, relocate, exchange and 2-opt* with best-accept strategy (for details, see the first part of this article). The proposed approach is a deterministic and entirely greedy one.

De Backer et al. (2000) test iterative improvement techniques within a Constraint Programming (CP) framework (for details, see part I of this survey). The improvement techniques are coupled to tabu search and GLS to avoid the search being trapped in local minima. The CP system is used only as background operator to check the validity of solutions and to speed up legality checks of improvement procedures. Four simple local search operators, namely 2-opt, relocate, exchange and 2-opt*, are used with the best-accept strategy to improve the solutions. The tabu search implementation is a simple one, involving only two tabu lists for storing recently removed and inserted arcs. The GLS implementation is similar to that of Kilby et al. (1999) described above. The initial solution is produced by the Savings method of Clarke and Wright (1964), followed by descent to a local minimum. GLS is found to be superior to the simple tabu search and a combined method, Guided Tabu Search, is concluded to perform slightly better than the simple tabu search and GLS on the long-haul benchmark problems (classes C2, R2 and RC2 of Solomon).

Liu and Shen (1999) propose a two-stage metaheuristic based on a new neighborhood structure focusing on the relationship between routes and nodes. In the construction phase, routes are constructed in a nested parallel manner by repeatedly estimating the lower bound for the unrouted customers. The seed customers are selected according to differences in time windows or so that they are geographically as dispersed as possible with regard to a previously chosen seed pair with the largest number of customers between them. The second step is to use these partial routes to service unrouted customers until no feasible insertion locations can be found. If an unrouted customer cannot be inserted into any route, five simple operations are used. These operations include insertion of one or two customers in

other routes in ejection chain manner (for details, see part I of this survey), and reordering the routes to make it possible to insert a new customer. In addition, a different set of unrouted customers is generated by exchanging routed and unrouted customers. During the route construction phase, a number of routes with low capacity utilization rates are eliminated. Once a feasible solution is constructed, λ-exchanges and simple reinsertions within the routes are used to improve solution quality. Finally, intra-route reinsertions that worsen the objective value are accepted to escape from local minima.

Gambardella et al. (1999) use an Ant Colony Optimization (ACO) approach (Dorigo et al., 1999) with a hierarchy of two cooperative artificial ant colonies. The first colony is used to minimize the number of vehicles while the second colony minimizes the total traveled distance. The two colonies cooperate through updating the best solution found, and in case the new best solution contains less vehicles, both colonies are reactivated with the reduced number of vehicles. The ACO is inspired by an analogy with real ant colonies foraging for food. In their search for food, ants mark the paths they travel by laying an aromatic essence called pheromone. The quantity of pheromone laid on the path depends on the length of the path and the quality of the food source. This pheromone provides information to other ants that are attracted to it. With time, paths leading to the more interesting food sources, i.e., close to the nest and with large quantities of food, become more frequented and are marked with larger amounts of pheromone. In the described VRPTW solution method, two measures are associated with each arc, the attractiveness N_{ij} and the pheromone trail T_{ij}. The tours are constructed using the nearest-neighbor heuristic with probabilistic rules, i.e., the next customer node to be inserted at the end of the current tour is not always the best according to N_{ij} and T_{ij}. During the search the pheromone trails are updated both locally and globally. The effect of local updating is to change dynamically the desirability of arcs: every time an ant uses an arc, the quantity of pheromone associated with this arc is decreased and the arc becomes less attractive. On the other hand, global updating is used to intensify the search in the neighborhood of the best solution computed. Each artificial ant constructs a separate feasible solution and the attractiveness N_{ij} is computed by taking into account the distance between customer nodes, the time window of the considered customer node and the number of times the considered customer node has not been inserted into the solution. In addition, the CROSS-exchanges of Taillard et al. (1997) are used to improve the quality of the feasible solutions.

Caseau et al. (1999b) hybridize several techniques, such as Limited Discrepancy Search (LDS), LNS, ejection chains (for details, see part I of this survey) and ejection trees. The initial solution is first constructed using the heuristic of Caseau and Laburthe (1999a). Ejection chains and ejection trees are then used to relocate most costful customers into other routes. The basic difference of ejection trees

compared to ejection chains is to remove more than one customer from a route while inserting only one customer. The LNS procedure used is the same as that proposed by Shaw (1998). The authors develop an algebra for these operators and test various combinations of different operators using several parameter values. A learning technique that automatically tunes an existing combination or discovers a new one is proposed. It is shown that automatic tuning yields a better solutions than hand-tuning with considerably less effort.

Rousseau et al. (2002) examine a variable neighborhood descent scheme introduced by Mladenovic and Hansen (1997) and new large neighborhood operators within a Constraint Programming framework. The first operator introduced is inspired by ideas from LNS of Shaw (1998). The algorithm removes first randomly a subset of customers with a bias towards customers generating the longest detour. A CP version of the GENI algorithm for the Traveling Salesman Problem with Time Windows (TSPTW) by Gendreau et al. (1998) is used to reinsert removed customers. The second operator introduced is called Naive Ejection Chains (Glover 1991 and 1992) and it is used to create the initial solution and diversify the search. In order to limit search space and prevent cycling, the first completed ejection chain is always accepted and each customer is allowed to be moved only once. The third proposed operator, SMART, removes a set of arcs instead of customers from the solution, creating an incomplete solution. The removed arcs can be either consecutive or randomly selected with a bias toward the longer arcs. This smaller routing problem is then solved either exactly by using the modified TSPTW model developed by Pesant et al. (1998) or, in case of a larger neighborhood size, by using LDS with a bounded number of discrepancies. The search oscillates between the two suggested operators (ejection chains are not considered here) to escape local minima. In the end, routes are reordered either exactly using the algorithm of Pesant et al. (1998) or, in case of longer routes, using the post-optimization part of the algorithm proposed by Pesant et al. (1997).

Bräysy (2001c) presents a new four-phase deterministic metaheuristic algorithm based on a modification of Variable Neighborhood Search (VNS) of Mladenovic and Hansen (1997). In the first phase, an initial solution is created using a construction heuristic that borrows its basic ideas from the studies of Solomon (1987) and Russell (1995). Routes are built one at a time in sequential fashion and after k customers have been inserted into the route, the route is reordered using Or-opt exchanges. Then a special route-elimination operator based on a new type of ejection chains (for details, see part I of this survey) is used to minimize the number of routes. In the third phase, the created solutions are improved in terms of distance using VNS oscillating between four new improvement procedures. These procedures are based on modifications to CROSS-exchanges of Taillard et al. (1997) and cheapest insertion heuristics. In the fourth phase, the objective function used by the local search operators is modified to consider also waiting time to escape from local minima.

Bräysy et al. (2002) continue the study of Bräysy (2001c) by introducing modifications to the construction and improvement heuristics, and by applying a new post-optimization technique based on Threshold Accepting (Dueck and Scheurer, 1990) that can be considered as deterministic modification of the simulated annealing. To be more precise, the reordering procedure is removed from the construction heuristic, and an extension of ejection chains that allows for infeasible solutions and removal of several customers from each route, is used in the second phase. In the distance optimization phase only modifications of CROSS-exchanges are used. The modifications include considering also inverting the order of the customers in the selected segments, and more insertion positions for segments. The post-optimization is based on a new inter-route exchange heuristic that combines the ideas of CROSS- and GENIUS-exchanges.

Anderson et al. (2000) propose an interactive scheme for VRPTW. The basic idea is to use a simple hill-climbing algorithm based on the relocation of n customers to find a local minimum. Then using visualization and interaction techniques, the human user identifies promising regions of the search space for the computer to explore, thus helping the search to escape from local minima. For example, the evaluation function for the moves may be edited by the user during the search and the user can assign priorities to specific customers to force some moves and choose between first-accept and best-accept strategies. Finally, a branch-and-bound algorithm is used to optimize each tour separately.

Ibaraki et al. (2002) introduce three methods for the vehicle routing problem with general time windows. The time window constraint for each customer is treated as a penalty function that can be non-convex and discontinuous as long as it is piecewise linear. After fixing the order of customers for a vehicle to visit, a dynamic programming method is used to determine the optimal start times to serve the customers so that the total penalty is minimized. The authors propose three metaheuristics to improve randomly generated initial solutions. Multi-start local search (MLS) creates and improves independently a number of initial solutions, and returns in the end the best solution obtained during the entire search. Iterated local search (ILS) is a variant of MLS, in which the initial solutions for local search procedure are generated by perturbing good solutions obtained during the search using random CROSS-exchanges. Adaptive multi-start local search (AMLS) keeps a set P of good solutions found in the previous search, and generates initial solutions for local search by combining parts of the solutions in P. The local search procedure uses four different neighborhoods in a variable neighborhood search manner, namely CROSS-exchange, 2-opt*, cyclic exchange and intra-route exchange based on Or-opt. The original cyclic transfer neighborhood and Or-opt exchanges described in the first part of this survey are modified so that they consider also reversing the order of the customers in the reinserted paths. The authors propose a number of strategies to search the neighborhoods efficiently, including ideas such as evaluating solutions in a specified order, time-oriented neighbor-list, using memory to

prevent searching unpromising regions of solution space and partial updating of the improvement graph for cyclic transfers.

The metaheuristic algorithms described above are compared in Table 5. We believe that Kontoravdis and Bard (1995) used rounded distance values after one decimal place. Thus their results are slightly better, for example, for the problem group C1 when compared to other approaches using real distance values. Only Liu and Shen (1999), Kilby et al. (1999) and Bräysy (2001c) propose deterministic methods. All other algorithms include some randomness in the search process. Most of the algorithms in Table 5 are implemented in C. However, Chiang and Russell (1996) and Liu and Shen (1999) use Fortran, Bräysy (2001c) uses Java and Kilby et al. (1999) and Tan et al. (2000) do not report the programming language used. According to our experience, Java is approximately 5–10 times slower than C, and there are not significant differences between C and Fortran regarding speed. Kilby et al. (1999) and Tan et al. (2000 and 2001c) consider only the total traveled distance in their objective function. For all other algorithms in Table 5, the primary optimization criterion is the number of vehicles. In most of the cases, the secondary criterion is the total distance of the routes. Only Potvin and Robillard (1995) and Chiang and Russell (1996) consider total duration of the routes as a secondary criterion, which must be taken into account in the comparison.

TABLE 5

Miscellaneous metaheuristic algorithms. For each algorithm the average results with respect to Solomon's benchmarks are reported. Notations CNV and CTD in the rightmost column indicate the cumulative number of vehicles and cumulative total distance over all 56 test problems.

Author	R1	R2	C1	C2	RC1	RC2	CNV/CTD
(1) Thangiah et al. (1994)	**12.33**	**3.00**	**10.00**	**3.00**	**12.00**	**3.38**	**418**
	1227.42	1005.00	830.89	640.86	1391.13	1173.38	58905
(2) Kontoravdis et al. (1995)	**12.58**	**3.09**	**10.00**	**3.00**	**12.63**	**3.50**	**427**
	1325.44	1164.27	827.3	589.65	1500.94	1414.21	64196
(3) Potvin et al. (1995)	**13.58**	**3.09**	**10.56**	**3.38**	**13.63**	**3.63**	**457**
	1539.4	1325.1	1237.2	875.6	1828.9	1578.9	78451
(4) Bachem et al. (1996)	**12.58**	**3.00**	—	—	**12.13**	**3.38**	—
	1392.0	1199.6			1501.6	1500.1	
(5) Chiang et al. (1996)	**12.50**	**2.91**	**10.00**	**3.00**	**12.38**	**3.38**	**422**
	1308.82	1166.42	909.80	666.30	1473.90	1393.70	64996
(6) Liu et al. (1999)	**12.17**	**2.82**	**10.00**	**3.00**	**11.88**	**3.25**	**412**
	1249.57	1016.58	830.06	591.03	1412.87	1204.87	59318
(7) Kilby et al. (1999)	**12.67**	**3.00**	**10.00**	**3.00**	**12.13**	**3.38**	**423**
	1200.33	966.56	830.75	592.24	1388.15	1133.42	57423
(8) Caseau et al. (1999b)	**12.17**	—	—	—	**12.00**	—	—
	1207.27				1356.62		
(9) Gambardella et al. (1999)	**12.00**	**2.73**	**10.00**	**3.00**	**11.63**	**3.25**	**407**
	1217.73	967.75	828.38	589.86	1382.42	1129.19	57525
(10) Tan et al. (2000)	**14.50**	**3.64**	**10.11**	**3.25**	**14.75**	**4.25**	**483**
	1420.12	1278.97	958.57	766.46	1648.77	1641.89	72194
(11) Anderson et al. (2000)	—	—	—	—	**11.63**	—	—

					1397		
(12) Tan et al. (2001c)	**13.10**	**4.60**	**10.00**	**3.30**	**12.70**	**5.60**	**470**
	1213.16	952.30	841.92	612.75	1415.62	1120.37	57799
(13) Bräysy (2001c)	**11.92**	**2.73**	**10.00**	**3.00**	**11.50**	**3.25**	**405**
	1222.12	975.12	828.38	589.86	1389.58	1128.38	57710
(14) Li et al. (2001)	**12.08**	**2.91**	**10.00**	**3.00**	**11.75**	**3.25**	**411**
	1215.14	953.43	828.38	589.86	1385.47	1142.48	57467
(15) Bent et al. (2001)	**11.92**	**2.73**	**10.00**	**3.00**	**11.50**	**3.25**	**405**
	1211.10	954.27	828.38	589.86	1384.17	1124.46	57273
(16) Rousseau et al. (2002)	**12.08**	**3.00**	**10.00**	**3.00**	**11.63**	**3.38**	**412**
	1210.21	941.08	828.38	589.86	1382.78	1105.22	56953
(17) Czech et al. (2002)	——	——	——	——	**11.50**	**3.25**	——
					1384.17	1119.49	
(18) Bräysy et al. (2002)	**12.00**	**2.73**	**10.00**	**3.00**	**11.50**	**3.25**	**406**
	1214.69	960.44	828.38	589.86	1389.20	1124.14	57422
(19) Ibaraki et al. (2002)	**11.92**	**2.73**	**10.00**	**3.00**	**11.50**	**3.25**	**405**
	1217.40	959.11	828.38	589.86	1391.03	1122.79	57444

(1) NeXT 25 MHz, –, 30 min. for all 8 methods, (2) Sun Sparc 10, 5 runs, 1.2 min., (3) Sun Sparc 2, –, 0.08 min., (4) Sun Sparc 10, –, (5) PC 486DX2/ 66 MHz, –, 2.4 min., (6) HP 9000/720, 3 runs, 20 min., (7) DEC Alpha, 3 runs, 48.3 min., (8) Pentium II 366 MHz, –, 5–30 min., (9) Sun Ultra Sparc 1 167 MHz, –,–, (10) Pentium II 266 MHz, –,–, (11) Pentium 500 MHz, –, 20 hours, (12) Pentium II 330 MHz, –, 4.6 min., (13) Pentium 200 MHz, 1 run, 82.5 min., (14) Pentium III 545 MHz, 3 runs, 30 min., (15) Sun Ultra 10, –,–., (16) Sun Ultra 10, 10 runs, 183.3 min., (17) 5×IBM RS/6000, –,–., (18) AMD 700 MHz, 30 runs, 2.7 min., (19) Pentium III 1 GHz, 1 run, 250 min.

According to Table 5, the best performing approaches in terms of solution quality seem to be Bent and Van Hentenryck (2001), Bräysy (2001c) and Ibaraki et al. (2002). Czech and Czarnas (2002) report excellent results to problem groups RC1 and RC2. If Kilby et al. (1999) and Tan et al. (2000 and 2001c) who focus on minimizing only total distance are not considered, the worst approach seems to be Potvin and Robillard (1995) utilizing neural networks. Basically the method of Potvin and Robillard (1995) is the same as the parallel construction heuristic of Potvin and Rousseau (1993). The neural network metaheuristic is only used to select the seed customers initializing the routes.

Generally the difference in number of vehicles is quite significant: it is about 5%, even if the worst approach by Potvin and Robillard (1995) is not considered. In terms of total traveled distance, the differences are much greater. For example, the difference between the methods proposed in Kontoravdis and Bard (1995) and Rousseau et al. (2002) in problem group RC2 is even 28%. However, if only the best performing approaches by Bent and Van Hentenryck (2001), Bräysy (2001c) and Ibaraki et al. (2002) are considered, the differences in CNV and CTD are only ≈0.8% and 0% respectively. It is quite interesting to note that even if Caseau and Laburthe (1999b) use a very sophisticated and intuitively appealing hybrid approach, its performance is not comparable with the other methods. Finally, the results of Anderson et al. (2000) suggest that the recent metaheuristic approaches outperform humans in designing efficient routes, though the differences are small.

Since many of the authors do not report the number of runs or the time consumption required to obtain the results in Table 5, final conclusions regarding which method performs best are impossible to

reach. Considering the information available, only methods of Kontoravdis and Bard (1995), Liu and Shen (1999), Bent and Van Hentenryck (2001), Bräysy (2001c), Ibaraki et al. (2002) and Bräysy et al. (2002) can be identified as Pareto optimal in terms of solution quality and time consumption. Another comparison with tabu searches and genetic algorithms is presented in Table 6 and Figure 1 of the next section, where only results for which computational effort is reported, are considered.

4. DISCUSSION

TO SUMMARIZE the results presented in Tables 2, 4 and 5, it appears that the ten metaheuristics showing the best performance are the ones by Gambardella et al. (1999), Homberger and Gehring (1999 and 2002), Gehring and Homberger (2001), Bent and Van Hentenryck (2001), Bräysy (2001c), Berger et al. (2001), Ibaraki et al. (2002), Bräysy et al. (2002) and Mester (2002). Six of these use so-called pools of solutions to memorize the best solutions found during the search. All the presented approaches use different local search techniques within the search and eight of the methods oscillate between different neighborhood structures. The neighborhood structures used seem to be typically quite small. Seven algorithms use modifications of traditional route construction heuristics to create an initial solution and nine methods create a set of several different initial solutions. All the methods use memory structures to facilitate the search and nine of them use special strategies or operators to reduce the number of routes. Two and five of the approaches employ ideas based on tabu search and evolutionary algorithms (genetic algorithms or evolution strategies), respectively. One uses ant colonies and two use multi-restart variable neighborhood search. All, except Bräysy (2001c) include some randomness in the search and three approaches allow infeasibilities during the search. Finally, four methods include different techniques to speed up the search. The method of Bent and Van Hentenryck seems to give the best output, and Bräysy et al. (2002) appears to be the fastest. The differences in CNV and CTD between the ten methods are quite small, 0.5% and 1.2%, respectively. Based on the above discussion, it seems that one cannot identify any special metaheuristic technique that would perform better than the others. Usage of memory and different traditional route construction and improvement techniques seems to work well. Keeping in memory a set of solutions found during the search and special strategies for reducing the number of routes are also important.

The efficiency of the described metaheuristic approaches is illustrated in Figure 1. Since the number of vehicles is often considered as the primary optimization criterion, we consider the Cumulative Number of Vehicles (CNV) as a good measure of the solution quality and robustness of a given approach. The closer the point is to the lower left corner in the Figure 1, the better the method is considered to be. That is, the objective is to achieve low CNV using as little CPU time as possible. We included in Figure 1 only approaches that report results for all 6 datasets of Solomon. Moreover, to be

able to analyze the computational effort, we consider here only results for which the time consumption and the number of runs are described. However, to clarify the figure, methods by Gehring and Homberger (2001), Bent and Van Hentenryck (2001) and Rousseau et al. (2002) are not considered here due to their large CPU times. To facilitate the comparison, the effect of different hardware is normalized to equal Sun Sparc 10 using Dongarra's (1998) factors, described in detail in Appendix I. In addition, if the reported results are the best ones over multiple experiments, we multiplied the computation times by this number to obtain the real computational effort.

From Figure 1 one can clearly see the development of various VRPTW algorithms over the past few years. In 1995, the best performing approach was that of Russell (1995). Correspondingly, in 1999 the Pareto optimal front in terms of solution quality and time consumption was formed by the methods of Russell (1995), Gehring and Homberger (1999), Liu and Shen (1999) and Homberger and Gehring (1999). At the moment, it seems that the only Pareto optimal approaches in terms of solution quality and time consumption are the local searches by Russell (1995), Bräysy (2001b) and Bräysy et al. (2002), and the VNS metaheuristic by Bräysy (2001c).

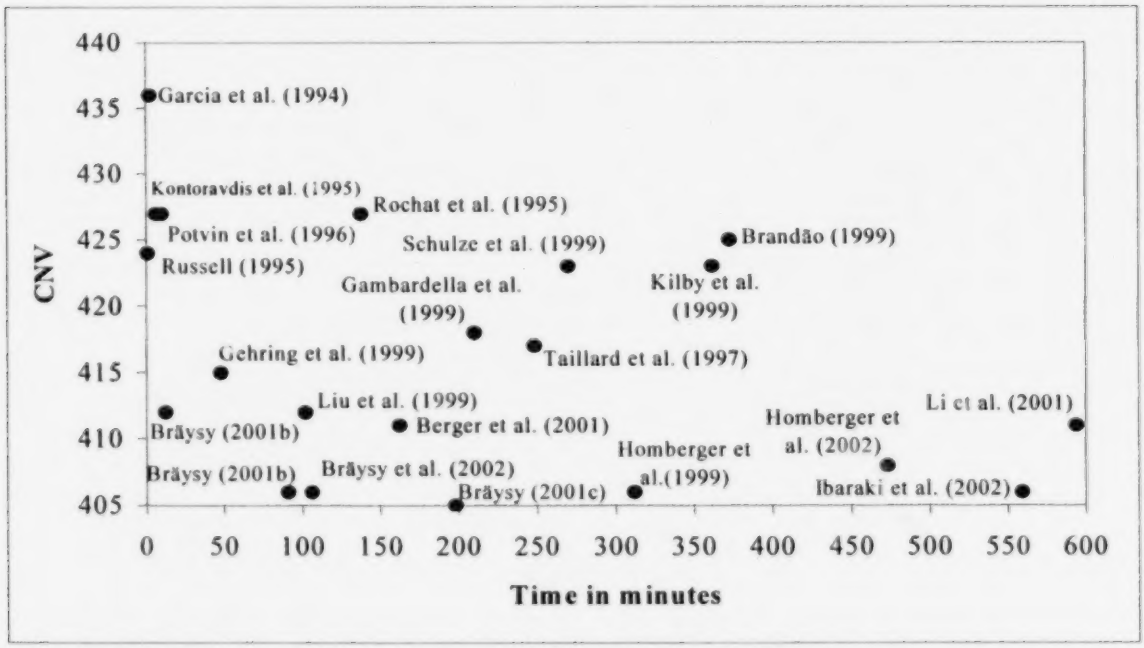

Figure 1. The efficiency of the described methods. The notation CNV refers to the cumulative number of vehicles over 56 test problems of Solomon (1987). Note that the computation times are normalized to equal Sun Sparc 10, using Dongarra's (1998) factors. Moreover, if the results are the best ones over multiple runs, the time consumption is multiplied by this number to illustrate the real computational burden. Local searches of Russell (1995) and Bräysy (2001b) are described in part I of the survey.

More detailed results are provided in Table 6. As in Figure 1, we consider here only results for which the time consumption and the number of runs are described. The computer, number of independent runs, and the CPU time used to obtain the reported results are described in the lower part of Table 6. The number of runs is greater than one only if the reported result is best over multiple executions of the given algorithm and the CPU time is reported only for a single run. Two CPU time values are described: the one reported by the authors and in the parenthesis the modified CPU time, computed in the same way as the values for Figure 1 above.

TABLE 6

Comparison of results obtained with limited computational effort for Solomon's benchmark problems.

Author	R1	R2	C1	C2	RC1	RC2	CNV/CTD
(1) Kontoravdis et al. (1995)	**12.58**	**3.09**	**10.00**	**3.00**	**12.63**	**3.50**	**427**
	1325.44	1164.27	827.3	589.65	1500.94	1414.21	64196
(2) Rochat et al. (1995)	**12.58**	**3.09**	**10.00**	**3.00**	**12.38**	**3.62**	**427**
	1197.42	954.36	828.45	590.32	1369.48	1139.79	57120
(3) Potvin et al. (1996)	**12.58**	**3.09**	**10.00**	**3.00**	**12.63**	**3.38**	**427**
	1294.7	1185.9	861.0	602.5	1465.0	1476.1	64679
(4) Taillard et al. (1997)	**12.33**	**3.00**	**10.00**	**3.00**	**11.90**	**3.38**	**417**
	1220.35	1013.35	828.45	590.91	1381.31	1198.63	58614
(5) Homberger et al. (1999)	**11.92**	**2.73**	**10.00**	**3.00**	**11.63**	**3.25**	**406**
	1228.06	969.95	828.38	589.86	1392.57	1144.43	57876
(6) Gehring et al. (1999)	**12.42**	**2.82**	**10.00**	**3.00**	**11.88**	**3.25**	**415**
	1198	947	829	590	1356	1144	56946
(7) Brandão (1999)	**12.58**	**3.18**	**10.00**	**3.00**	**12.13**	**3.50**	**425**
	1205	995	829	591	1371	1250	58562
(8) Schulze et al. (1999)	**12.50**	**3.09**	**10.00**	**3.00**	**12.25**	**3.38**	**423**
	1268.42	1055.90	828.94	589.93	1396.07	1308.31	60651
(9) Gambardlpella et al.	**12.38**	**3.00**	**10.00**	**3.00**	**11.92**	**3.33**	**418**
(1999)	1210.83	960.31	828.38	591.85	1388.13	1149.28	57583
(10) Kilby et al. (1999)	**12.67**	**3.00**	**10.00**	**3.00**	**12.13**	**3.38**	**423**
	1200.33	966.56	830.75	592.24	1388.15	1133.42	57423
(11) Liu et al (1999)	**12.17**	**2.82**	**10.00**	**3.00**	**11.88**	**3.25**	**412**
	1249.57	1016.58	830.06	591.03	1412.87	1204.87	59318
(12) Bräysy (2001c)	**11.92**	**2.73**	**10.00**	**3.00**	**11.50**	**3.25**	**405**
	1222.12	975.12	828.38	589.86	1389.58	1128.38	57710
(13) Gehring et al. (2001)	**12.00**	**2.73**	**10.00**	**3.00**	**11.50**	**3.25**	**406**
	1217.57	961.29	828.63	590.33	1395.13	1139.37	57641
(14) Li et al. (2001)	**12.08**	**2.91**	**10.00**	**3.00**	**11.75**	**3.25**	**411**
	1215.14	953.43	828.38	589.86	1385.47	1142.48	57467
(15) Bent et al. (2001)	**12.17**	**2.73**	**10.00**	**3.00**	**11.63**	**3.25**	**409**
	1203.84	980.31	828.38	589.86	1379.03	1158.91	57707
(16) Berger et al. (2001)	**12.17**	**2.73**	**10.00**	**3.00**	**11.88**	**3.25**	**411**
	1251.40	1056.59	828.50	590.06	1414.86	1258.15	60200
(17) Rousseau et al. (2002)	**12.08**	**3.00**	**10.00**	**3.00**	**11.63**	**3.38**	**412**
	1210.21	941.08	828.38	589.86	1382.78	1105.22	56953
(18) Homberger et al. (2002)	**12.08**	**2.82**	**10.00**	**3.00**	**11.50**	**3.25**	**408**
	1211.67	950.72	828.45	589.96	1395.93	1135.09	57422
(19) Bräysy et al. (2002)	**12.00**	**2.73**	**10.00**	**3.00**	**11.50**	**3.25**	**406**
	1220.20	970.38	828.38	589.86	1398.76	1139.37	57796
(20) Ibaraki et al. (2002)	**11.92**	**2.73**	**10.00**	**3.00**	**11.63**	**3.25**	**406**
	1220.02	961.64	828.38	589.86	1378.72	1132.17	57480

(1) Sun Sparc 10, 5 runs, 1.2 (6.0) min., (2) Silicon Graphics 100 MHz, 1 run, 92.2 (138) min., (3) Sun Sparc 10, 1 run, 9.8 (9.8) min., (4) Sun Sparc 10, 1 run, 248 (248) min., (5) Pentium 200 MHz, 10 runs, 13 (312) min., (6) 4×Pentium 200 MHz, 1 run, 5 (48) min., (7) Pentium 200, 4 runs, 38.9 (373) min., (8) Motorola PowerPC 604, 5 runs, 8.3 (270) min., (9) Sun Ultrasparc 1, 1 run 30 (210) min., (10) DEC Alpha, 3 runs, 48.3 (362) min., (11) HP 9000/720, 3 runs, 20 (102) min., (12) Pentium 200 MHz, 1 run, 82.5 (198) min., (13) 4×Pentium 400 MHz, 5 runs, 13.5 (1458) min., (14) Pentium III 545 MHz, 3 runs, 30 (594) min., (15) Sun Ultra 10, 5 runs, 30 (1095) min., (16) Pentium 400 MHz, 1 run, 30 (162) min., (17) Sun Ultra 10, 10 runs, 183.3 (13 381) min., (18) Pentium 400 MHz, 5 runs, 17.5 (473) min., (19) AMD 700 MHz, 3 runs, 2.6 (106) min., (20) Pentium III 1 GHz, 1 run, 33 (559) min.

According to Table 6, the algorithms by Homberger and Gehring (1999), Gehring and Homberger (2001), Bräysy (2001c), Ibaraki et al. (2002) and Bräysy et al. (2002) show the best overall performance. Regarding individual problem groups, it seems that practically all approaches yield excellent results for problem groups C1 and C2, having customers located in geographical clusters. Ibaraki et al. (2002) provides the best output for group R1, Gehring and Homberger (2001) report the best results for group R2 and Bräysy (2001c) for RC1 and RC2. In general, the differences between the recent results, reported in years 2001 and 2002, are small, varying within 2%. Here one must note that only Kilby et al. (1999), Liu and Shen (1999) and Bräysy (2001c) describe deterministic methods. All other approaches in Table 6 are non-deterministic, and the presented results are often the best ones over several runs. Thus, there is no guarantee for obtaining similar results every time as is the case with deterministic methods. For detailed best-known results to Solomon's benchmarks, we refer the reader to http://www.top.sintef.no/.

During the last few years, several authors have also tested their algorithms with other benchmark problems, such as two real-life problems of Russell (1995) and the extended Solomon's benchmark problems by Gehring and Homberger (1999). Comparisons regarding Russell's test cases can be found in Bräysy et al. (2002), and at the moment the best results are reported in Gehring and Homberger (2001), Bräysy (2001c) and Bräysy et al. (2002). We are aware of nine papers tackling the extended Solomon's benchmark problems: Gehring and Homberger (1999 and 2001), Bräysy (2001c), Li and Lim (2001), Bent and Van Hentenryck (2001), Mester (2002), Homberger and Gehring (2002), Le Bouthillier and Crainic (2002) and Bräysy et al. (2002). Bräysy (2001c) reports the best average performance for 200 and 400-customer data sets, and is also faster than the competing approaches. The differences are however small. On the larger benchmarks Gehring and Homberger (2001) appear to perform best. On the other hand, most of the best-known solutions are reported in Mester (2002) and Bräysy et al. (2002). For details, see Bräysy et al. (2002) and http://www.top.sintef.no/.

5. CONCLUSIONS

NP-HARDNESS of the VRPTW requires heuristic solution strategies for most real-life instances. In the previous sections, we have comprehensively surveyed the remarkable evolution of metaheuristic VRPTW methodologies. Currently, algorithms by Bent and Van Hentenryck (2001), Bräysy (2001c), Berger et al. (2001), Homberger and Gehring (2002) and Ibaraki et al. (2002) appear to achieve the best robustness. The quality of the solutions obtained with different metaheuristic techniques is often much better compared to traditional construction heuristics and local search algorithms. At the same time, metaheuristics require more CPU time and are more complex to implement and calibrate. This might prove quite significant in real, practical settings. Another issue of concern when considering the choice of a solution approach for real-life applications is that of flexibility, i.e., how well various approaches can handle the notoriously "dirty" additional constraints that almost always clutter practical instances. In general, local search and metaheuristic techniques perform well in this respect, but some methods may prove more effective in their treatment of complicating constraints. This is the case, for instance, of the Constraint Programming-based approach of Rousseau et al. (2002) that was specifically developed with this objective in mind. The final choice of the methodology to apply in any given setting thus requires a careful analysis in order to properly balance the different criteria that need to be considered.

We believe that, in the future, the research on faster metaheuristics, incorporating various sophisticated speed up techniques will intensify. Parallel implementations, hybridizations of different heuristics and with exact techniques (and also constraint programming), very large scale neighborhoods, and different adaptive memory-based approaches sound also promising concepts. Possible future trends may also include tailored solution approaches based on careful analysis of the problem at hand, and learning during the search process. On the other hand, as pointed out in Cordeau et al. (2002), the research on simpler and more flexible, yet effective, metaheuristics will also increase.

ACKNOWLEDGEMENTS

THIS WORK WAS partially supported by the Emil Aaltonen Foundation, the Canadian Natural Science and Engineering Research Council, Liikesivistysrahasto and Volvo Foundations, and the TOP project funded by the Research Council of Norway. This support is gratefully acknowledged.

REFERENCES

Aarts, J., H.M. Korst and P.J.M. Van Laarhaven (1997), "Simulated Anne ling", in *Local Search in Combinatorial Optimization*, E. Aarts and J.K. Lenstra (eds), 91–120, John Wiley & Son ichester.

Alander, J.T. (2000), "An Indexed Bibliography of Genetic Algorithms in Operations Research", Technical report No. 94-1-OR, University of Vaasa, Finland. Available via anonymous `ftp:` site `ftp.uwasa.fi` directory `cs/report94-1` file `gaORbib.ps.Z`.

Anderson, D., E. Anderson, N. Lesh, J. Marks, B. Mirtich, D. Ratajczak and K. Ryall (2000), "Human-Guided Simple Search", Working Paper, Mitsubishi Electric Research Laboratory, Cambridge, U.S.A.

Bachem, A., W. Hochstättler and M. Malich (1996), "The Simulated Trading Heuristic for Solving Vehicle Routing Problems", *Discrete Applied Mathematics* 65, 47–72.

Badeau, P., M. Gendreau, F. Guertin, J.-Y. Potvin and E. Taillard (1997), "A Parallel Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows", *Transportation Research – C* : 109–122.

Beasley, J.E (1990), "A Lagrangian Heuristic for Set Covering Problems", *Naval Research Logistics* 37, 151–164.

Bent, R. and P. Van Hentenryck (2001), "A Two-Stage Hybrid Local Search for the Vehicle Routing Problem with Time Windows", Technical Report CS-01-06, Department of Computer Science, Brown University.

Benyahia, I. and J.-Y. Potvin (1995), "Generalization and Refinement of Route Construction Heuristics using Genetic Algorithms", in *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*, 39–43, IEEE Service Center, Piscataway, U.S.A.

Berger, J., M. Salois and R. Begin (1998), "A Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows", *Lecture Notes in Artificial Intelligence* 1418, 114–127.

Berger, J., M. Barkaoui and O. Bräysy (2001), "A Route-directed Hybrid Genetic Approach for the Vehicle Routing Problem with Time Windows", To Appear in *INFOR*.

Blanton, J.L. and R.L. Wainwright (1993), "Multiple Vehicle Routing with Time and Capacity Constraints using Genetic Algorithms", in *Proceedings of the Fifth International Conference on Genetic Algorithms*, S. Forrest (ed), 452–459, Morgan Kaufmann Publishing, San Francisco.

Brandão, J. (1999), "Metaheuristic for the Vehicle Routing Problem with Time Windows", in *Meta-heuristics – Advances and Trends in Local Search Paradigms for Optimization*, S. Voss, S. Martello, I.H. Osman and C. Roucairol (eds), 19–36, Kluwer Academic Publishers, Boston.

Bräysy, O. (2001a), Local Search and Variable Neighborhood Search Algorithms for the Vehicle Routing Problem with Time Windows, Doctoral Dissertation, University of Vaasa, Finland.

Bräysy, O. (2001b), "Fast Local Searches for the Vehicle Routing Problem with Time Windows", To Appear in *INFOR*.

Bräysy, O. (2001c), "A Reactive Variable Neighborhood Search for the Vehicle Routing Problem with Time Windows", To Appear in *INFORMS Journal on Computing*.

Bräysy, O. and M. Gendreau (2002a), "Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms", To Appear in *Transportation Science*.

Bräysy, O. and M. Gendreau (2002b), "Tabu Search Heuristics for the Vehicle Routing Problem with Time Windows", To Appear in *TOP*.

Bräysy, O. and M. Gendreau (2002c), "Evolutionary Algorithms for the Vehicle Routing Problem with Time Windows", Working paper, SINTEF Applied Mathematics, Department of Optimization, Norway.

Bräysy, O., G. Hasle and W. Dullaert (2002), "A Fast Local Search Algorithm for the Vehicle Routing Problem with Time Windows", Working paper, SINTEF Applied Mathematics, Department of Optimization, Norway.

Carlton, W.B. (1995), A Tabu Search Approach to the General Vehicle Routing Problem, Ph.D. thesis, University of Texas, Austin, U.S.A.

Caseau, Y. and F. Laburthe (1999a), "Heuristics for Large Constrained Vehicle Routing Problems", *Journal of Heuristics* 5, 281–303.

Caseau, Y., F. Laburthe and G. Silverstein (1999b), "A Meta-heuristic Factory for Vehicle Routing Problems", in *Principles and Practice of Constraint Programming– CP'99, Lecture Notes in Computer Science*, J. Jaffar (ed), 144–158, Springer-Verlag, New York.

Chiang, W.C. and R.A. Russell (1996), "Simulated Annealing Metaheuristics for the Vehicle Routing Problem with Time Windows", *Annals of Operations Research* 63, 3–27.

Chiang, W.C. and R.A. Russell (1997), "A Reactive Tabu Search Metaheuristic for the Vehicle Routing Problem with Time Windows", *INFORMS Journal on Computing* 9, 417–430.

Christofides, N. and J. Beasley (1984), "The Period Routing Problem", *Networks* 14, 237–246.

Clarke, G. and J.W. Wright (1964), "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points", *Operations Research* 12, 568–581.

Cordeau, J.-F., G. Laporte, A. Mercier (2001), "A Unified Tabu Search Heuristic for Vehicle Routing Problems with Time Windows", *Journal of the Operational Research Society* 52, 928–936.

Cordeau, J.-F., M. Gendreau, G. Laporte, J.-Y. Potvin and F. Semet (2002), "A Guide to Vehicle Routing Heuristics", *Journal of the Operational Research Society* 53, 512–522.

Czech, Z. and P. Czarnas (2002), "Parallel Simulated Annealing for the Vehicle Routing Problem with Time Windows", in *Proceedings of 10th Euromicro Workshop on Parallel Distributed and Network-Based Processing*, 376–383, Canary Islands, Spain.

De Backer, B. and V. Furnon (1997), "Meta-heuristics in Constraint Programming Experiments with Tabu Search on the Vehicle Routing Problem", in *Proceedings of the Second International Conference on Metaheuristics* (MIC'97), 1–14, Sophia Antipolis, France.

De Backer, B., V. Furnon, P. Kilby, P. Prosser and P. Shaw (2000), "Solving Vehicle Routing Problems Using Constraint Programming and Metaheuristics", *Journal of Heuristics* 6, 501–523.

Dongarra, J. (1998), "Performance of Various Computers Using Standard Linear Equations Software", Report CS-89-85, Department of Computer Science, University of Tennessee, U.S.A.

Dorigo, M., G. Di Caro and L.M. Gambardella (1999), "Ant Algorithms for Discrete Optimization", *Artificial Life* 5, 137–172.

Dueck, G. and T. Scheurer (1990), "Threshold Accepting: A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing", *Journal of Computational Physics* 90, 161–175.

Faigle, U. and W. Kern (1992), "Some Convergence Results for Probabilistic Tabu Search", *ORSA Journal on Computing* 4, 32–37.

Fogel, D.B. (1995), *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, IEEE Press, New York.

Fox, B.L. (1993), "Integrating and Accelerating Tabu Search, Simulated Annealing and Genetic Algorithms", *Annals of Operations Research* 41, 47–67.

Gambardella, L.M., E. Taillard and G. Agazzi (1999), "MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows", in *New Ideas in Optimization*, D. Corne, M. Dorigo and F. Glover (eds), 63–76, McGraw-Hill, London.

Garcia, B.-L., J.-Y. Potvin and J.-M. Rousseau (1994), "A Parallel Implementation of the Tabu Search Heuristic for Vehicle Routing Problems with Time Window Constraints", *Computers & Operations Research* 21, 1025–1033.

Gehring, H. and J. Homberger (1999), "A Parallel Hybrid Evolutionary Metaheuristic for the Vehicle Routing Problem with Time Windows", in *Proceedings of EUROGEN99*, K. Miettinen, M. Mäkelä and J. Toivanen (eds), 57–64, University of Jyväskylä, Finland.

Gehring, H. and J. Homberger (2001), "Parallelization of a Two-Phase Metaheuristic for Routing Problems with Time Windows", *Asia-Pacific Journal of Operational Research* 18, 35–47.

Gendreau, M., A. Hertz and G. Laporte (1992), "A New Insertion and Postoptimization Procedures for the Traveling Salesman Problem", *Operations Research* 40, 1086–1093.

Gendreau, M., A. Hertz and G. Laporte (1994), "A Tabu Search Heuristic for the Vehicle Routing Problem", *Management Science* 40, 1276–1290.

Gendreau, M., A. Hertz, G. Laporte and M. Stan (1998), "A Generalized Insertion Heuristic for the Traveling Salesman Problem with Time Windows", *Operations Research* 46, 330–335.

Gendreau, M. (2003), "An Introduction to Tabu Search", in *Metaheuristic Handbook*, F. Glover and G.A. Kochenberger (eds.), Kluwer Academic Publishers, Boston (forthcoming).

Gillett, B. and L.R. Miller (1974), "A Heuristic Algorithm for the Vehicle Dispatch Problem", *Operations Research* 22, 340–349.

Glover, F. (1986), "Future Paths for Integer Programming and Links to Artificial Intelligence", *Computers and Operations Research* 13, 533–549.

Glover, F. (1989), "Tabu Search – Part I", *Journal on Computing* 1, 190–206.

Glover, F. (1990), "Tabu Search – Part II", *Journal on Computing* 2, 4–32.

Glover, F. (1991), "Multilevel Tabu Search and Embedded Search Neighborhoods for the Traveling Salesman Problem", Working Paper, College of Business & Administration, University of Colorado, Boulder.

Glover, F. (1992), "New Ejection Chain and Alternating Path Methods for Traveling Salesman Problems", in *Computer Science and Operations Research: New Developments in Their Interfaces*, O. Balci, R. Sharda, and S. Zenios (eds), 449–509, Pergamon Press, Oxford.

Glover, F. and M. Laguna (1997), *Tabu Search*, Kluwer Academic Publishers, Boston.

Goldberg, D. (1989), *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley Publishing Company Inc, New York.

Golden, B. L. and W. R. Stewart (1985), "Empirical Analysis of Heuristics", in *The Traveling Salesman Problem*, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys (eds), 207–249, John Wiley & Sons, Chichester.

Hertz, A., E. Taillard and D. de Werra (1997), "Tabu Search", in *Local Search in Combinatorial Optimization*, E. Aarts and J. K. Lenstra (eds), 121–136, John Wiley & Sons, Chichester.

Holland, J.H. (1975), *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor.

Homberger, J. and H. Gehring (1999), "Two Evolutionary Meta-heuristics for the Vehicle Routing Problem with Time Windows", *INFOR* 37, 297–318.

Homberger, J. and H. Gehring (2002), "A Two-Phase Hybrid Metaheuristic for the Vehicle Routing Problem with Time Windows", To Appear in *European Journal of Operational Research.*

Hopfield, J.J. and D.W. Tank (1985), "Neural Computation of Decisions in Optimization Problems", *Biological Cybernetics* 52, 141–152.

Ibaraki, T., S. Imahori, M. Kubo, T. Masuda, T. Uno and M. Yagiura (2002), "Effective Local Search Algorithms for the Vehicle Routing Problem with General Time Window Constraints", Working paper, Department of Applied Mathematics and Physics, Kyoto University, Japan.

Jong De, K.A. (1975), An Analysis of the Behavior of a Class of Genetic Adaptive Systems, Ph.D. thesis, University of Michigan, U.S.A.

Jung, S. and B.-R. Moon (2002), "A Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows", in *Proceedings of Genetic and Evolutionary Computation Conference*, 1309–1316, Morgan Kaufmann, San Francisco.

Kilby, P., P. Prosser and P. Shaw (1999), "Guided Local Search for the Vehicle Routing Problem with Time Windows", in *META-HEURISTICS Advances and Trends in Local Search Paradigms for Optimization*, S. Voss, S. Martello, I.H. Osman and C. Roucairol (eds), 473–486, Kluwer Academic Publishers, Boston.

Kirkpatrick, S., C.D. Gelatt and P.M. Vecchi (1983), "Optimization by Simulated Annealing", *Science* 220, 671–680.

Kohonen, T. (1988), *Self-Organization and Associative memory*, Springer-Verlag, Berlin.

Kontoravdis, G.A. and J.F. Bard (1995), "A GRASP for the Vehicle Routing Problem with Time Windows", *Journal on Computing* 7, 10–23.

Lau, H.C., Y. F. Lim and Q. Liu (2000), "Diversification of Neighborhood via Constraint-based Local Search and Its Application to VRPTW", Working paper, School of Computing, National University of Singapore.

Lau, H.C., M. Sim and K.M. Teo (2002), "Vehicle Routing Problem with Time Windows and a Limited Number of Vehicles", To Appear in *European Journal of Operational Research.*

Le Bouthillier, A. and T.G. Crainic (2002), "Co operative Parallel Metaheuristic for Vehicle Routing with Time Windows", presented at Optimization Days, Montreal, Canada, May 6-8th.

Li, H., A. Lim, J. Huang (2001), "Local Search with Annealing-like Restarts to solve the VRPTW", To Appear in *European Journal of Operational Research.*

Li, H. and A. Lim (2001), "Large Scale Time-Constrained Vehicle Routing Problems: A General Metaheuristic Framework with Extensive Experimental Results", To Appear in *AI Review.*

Liu, F.-H. and S.-Y. Shen (1999), "A Route-neighborhood-based Metaheuristic for Vehicle Routing Problem with Time Windows", *European Journal of Operations Research* 118, 485–504.

Mester, D. (2002), "An Evolutionary Strategies Algorithm for Large Scale Vehicle Routing Problem with Capacitate and Time Window Restrictions", Working Paper, Institute of Evolution, University of Haifa, Israel.

Metropolis, W., A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller (1953), "Equation of the State Calculations by Fast Computing Machines", *Journal of Chemical Physics* 21, 1087–1092.

Mladenovic, N. and P. Hansen (1997), "Variable Neighborhood Search", *Computers & Operations Research* 24, 1097–1100.

Mühlenbein, H. (1997), "Genetic Algorithms", in *Local Search in Combinatorial Optimization*, E. Aarts and J. K. Lenstra (eds), 137–172, John Wiley & Sons, Chichester.

Or, I. (1976), Traveling Salesman-Type Combinatorial Problems and their Relation to the Logistics of Regional Blood Banking, Ph.D. thesis, Northwestern University, Illinois.

Pesant, G., M. Gendreau and J.M. Rousseau (1997), "GENIUS-CP: a Generic Vehicle Routing Algorithm", in *Principles and Practice of Constraint Programming – CP97, Lecture Notes in Computer Science*, G. Smolka (ed), 420–434, Springer-Verlag, New York.

Pesant, G., M. Gendreau, J.-Y. Potvin and J.-M. Rousseau (1998), "An Exact Constraint Logic Programming Algorithm for the Traveling Salesman Problem with Time Windows", *Transportation Science* 32, 12–29.

Potvin, J.-Y. and J.-M. Rousseau (1993), "A Parallel Route Building Algorithm for the Vehicle Routing and Scheduling Problem with Time Windows", *European Journal of Operational Research* 66, 331–340.

Potvin, J.-Y. and C. Robillard (1995), "Clustering for Vehicle Routing with a Competitive Neural Network", *Neurocomputing* 8, 125–139.

Potvin, J.-Y. and S. Bengio (1996), "The Vehicle Routing Problem with Time Windows Part II: Genetic Search", *INFORMS Journal on Computing* 8, 165–172.

Potvin, J.-Y., T. Kervahut, B.L. Garcia and J.-M. Rousseau (1996), "The Vehicle Routing Problem with Time Windows Part I: Tabu Search", *INFORMS Journal on Computing* 8, 157–164.

Potvin, J.-Y., D. Dubé and C. Robillard (1996), "A Hybrid Approach to Vehicle Routing Using Neural Networks and Genetic Algorithms", *Applied Intelligence* 6, 241-252.

Rechenberg, I. (1973), *Evolutionsstrategie*, Fromman-Holzboog, Stuttgart.

Rochat, Y. and E. Taillard (1995), "Probabilistic Diversification and Intensification in Local Search for Vehicle Routing", *Journal of Heuristics* 1, 147–167.

Rousseau, L.-M., M. Gendreau and G. Pesant (2002), "Using Constraint-Based Operators to Solve the Vehicle Routing Problem with Time Windows", *Journal of Heuristics* 8, 43–58.

Russell, R.A. (1995), "Hybrid Heuristics for the Vehicle Routing Problem with Time Windows", *Transportation Science* 29, 156–166.

Schulze, J. and T. Fahle (1999), "A Parallel Algorithm for the Vehicle Routing Problem with Time Window Constraints", *Annals of Operations Research* 86, 585–607.

Schwefel, H.-P. (1977), *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*, Birkhäuser, Basel.

Shaw, P. (1998), "Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems", in *Principles and Practice of Constraint Programming – CP98, Lecture Notes in Computer Science*, M. Maher and J.-F. Puget (eds), 417–431, Springer-Verlag, New York.

Solomon, M.M. (1987), "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints", *Operations Research* 35, 254–265.

Taillard, E., P. Badeau, M. Gendreau, F. Guertin and J.-Y. Potvin (1997), "A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows", *Transportation Science* 31, 170–186.

Tan, K.C., L.H. Lee and K.Q. Zhu (2000), "Heuristic Methods for Vehicle Routing Problem with Time Windows", in *Proceedings of the 6th International Symposium on Artificial Intelligence & Mathematics*, Ft. Lauderdale, Florida.

Tan, K.C., L. H. Lee and K. Ou (2001a), "Hybrid Genetic Algorithms in Solving Vehicle Routing Problems with Time Window Constraints", *Asia-Pacific Journal of Operational Research* 18, 121–130.

Tan, K.C., T.H. Lee, K. Ou and L.H. Lee (2001b), "A Messy Genetic Algorithm for the Vehicle Routing Problem with Time Window Constraints", in *Proceedings of the 2001 Congress on Evolutionary Computation*, 679–686, IEEE Service Center, Pistacaway.

Tan, K.C., L.H. Lee and K. Ou (2001c), "Artificial Intelligence Heuristics in Solving Vehicle Routing Problems with Time Window Constraints", *Engineering Applications in Artificial Intelligence* 14, 825–837.

Thangiah, S.R., K.E. Nygard and P.L. Juell (1991), "GIDEON: A Genetic Algorithm System for Vehicle Routing with Time Windows", in *Proceedings of Seventh IEEE Conference on Artificial Intelligence Applications*, 322–328, IEEE Computer Society Press, Los Alamitos.

Thangiah, S.R., I. Osman and T. Sun (1994), "Hybrid Genetic Algorithm, Simulated Annealing and Tabu Search Methods for Vehicle Routing Problems with Time Windows", Working Paper UKC/IMS/OR94/4, Institute of Mathematics and Statistics, University of Kent, Canterbury.

Thangiah, S.R. (1995a), "Vehicle Routing with Time Windows Using Genetic Algorithms", in *Application Handbook of Genetic Algorithms: New Frontiers, Volume II*, L. Chambers (ed), 253–277, CRC Press, Boca Raton.

Thangiah, S.R. (1995b), "An Adaptive Clustering Method Using a Geometric Shape for Vehicle Routing Problems with Time Windows", in *Proceedings of the 6th International Conference on Genetic Algorithms*, L.J. Eshelman (ed), 536–543, Morgan Kaufmann, San Francisco.

Thangiah, S.R., I.H. Osman, R. Vinayagamoorthy and T. Sun (1995), "Algorithms for the Vehicle Routing Problems with Time Deadlines", *American Journal of Mathematical and Management Sciences* 13, 323–355.

Voudouris, C. (1997), Guided Local Search for Combinatorial Problems, Ph.D. thesis, Department of Computer Science, University of Essex, Colchester, UK Voudouris, C. and E. Tsang (1998), "Guided Local Search", *European Journal of Operations Research* 113, 80–119.

Wee Kit, H., J. Chin and A. Lim (2001), "A Hybrid Search Algorithm for the Vehicle Routing Problem with Time Windows", *International Journal on Artificial Intelligence Tools* 10, 431–449.

APPENDIX 1

Dongarra's (1998) factors for the approaches described in Figure 1 and Table 6.

Computer:	Mflops/s
16 × Meiko T-800	7
PC 486 / 66 MHz	1.48
Sun Sparc 10	10
Silicon Graphics 100 MHz	15
Sun Ultra EnterPrise 450	44
8× Motorola PowerPC 604	65
Sun Ultra Sparc 1 167 MHz	70
Sun Ultra 10	73
DEC Alpha	25
HP 9000/720	17
Pentium 200 MHz	24
Pentium 366 MHz	48
Pentium 400 MHz	54
Pentium III 545 MHz	66
AMD 700 MHz	136
Pentium III 1 GHz	168